Softwarepraktikum 2003

MathServer

Projektdokumentation
(Teilbeleg 1)

Teamleiter: Thomas Weise
Mitglieder des Projektteams:
Roland Fischer
René Kreiner
Rico Roßberg
Thomas Ziegs

Praktikumsbetreuer : Dipl.-Inf. Lutz Neugebauer

Chemnitz, den 19.05.2003

1 Spezifikation der funktionellen Anforderungen

1.1 Produktbeschreibung

Das Mathematikprogramm MathServer für den Einsatz in Hochschulrechenzentren unterstützt bei der Bewältigung von Rechenaufgaben. Dazu gehören einfache Grundrechenoperationen bis hin zu komplizierteren mathematischen Berechnungen. Das System muss leicht erweiterbar sein, um es später auch zusammen mit anderen Betriebssystemen betreiben zu können.

Nutzungsumgebung: Für die eigentlichen Berechnungen wird ein leistungsstarker Großrechner verwendet. Die Benutzer arbeiten jedoch an leistungsschwächeren Workstations. Diese sind über ein lokales Netzwerk mit dem Großrechner verbunden.

Nutzergruppen: Das System wird von Studenten und dem wissenschaftlichen Personal (Benutzer) verwendet. Die Serverkomponente wird von einem Administrator am Großrechner gewartet.

1.2 Funktionelle Anforderungen

1.2.1 Umgebungsmodell

Auf Grund der Projektbeschreibung muss die Applikation in einen Klienten und einen Serverteil untergliedert werden, damit sie die gestellten Anforderungen im Bezug auf Netzwerkfähigkeit erfüllen kann.

An dieser Stelle wollen wir einige Begriffe definieren, die im Nachfolgenden häufiger verwendet werden.

Tabelle 1.1

Begriffe	**Bedeutung**
Math-Server	Der Serverteil des Programms.
Math-Client	Der Kliententeil des Programms.
Matheaufgabe	Mathematische Formel mit Zahlenwerten die genau zu einem Ergebnis führt (ähnlich Taschenrechner)
Matheaufgabe_oZ	Matheaufgabe ohne Zwischenergebnisanforderung
Matheaufgabe_mZ	Matheaufgabe mit Forderung ausgewählter Zwischenergebnisse
Ergebnis	Ergebnis der Matheaufgabe
Ergebnis_oZ	Endergebnis ohne Zwischenergebnisausgabe
Ergebnis_mZ	Endergebnis mit Ausgabe der ausgewählten Zwischenergebnisse
ungültige Daten	Nicht korrekt formulierte Matheaufgabe, nicht lösbare Matheaufgabe (allgemein und für unsere Applikation)
Error	Fehlerrückmeldung

1.2.1.1 Ereignistabelle

Bei unserem Projekt handelt es sich gemäß der der Produktbeschreibung um eine verteilte Anwendung. Der Serverteil (Math-Server) ist nicht gezwungenermaßen an den Kliententeil (Math-Client) gebunden. Er kann theoretisch auch mit anderen Klienten betrieben werden. Daher stellen wir an dieser Stelle die Ereignistabellen von sowohl dem Math-Server als auch des Math-Clients dar. Für den normalen Betrieb mit dem mitgelieferten Klienten kann die Ereignistabelle des Math-Clients als Ereignistabelle des Gesamtsystems betrachtet werden.
Es treten keine Ereignisse auf, die durch den Administrator ausgelöst werden, da dieser nur bei Installation und Wartung der Applikation in Erscheinung tritt.

Tabelle 1.2

Ereignistabelle des Math-Servers			
Nr.	**Ereignis**	**Auslöser**	**Antwort**
1S.	Math-Client übermittelt Rechenaufgabe (ohne Zwischenergebnisse anzufordern)	Matheaufgabe_oZ	Ergebnis_oZ
2S.	Math-Client übermittelt Rechenaufgabe (ausgewählte Zwischenergebnisse gefordert)	Matheaufgabe_mZ	Ergebnis_mZ

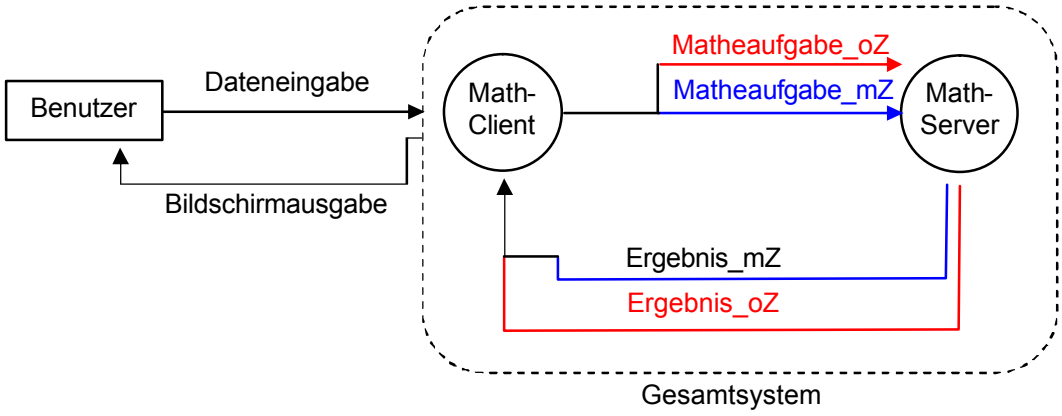
Tabelle 1.3

Ereignistabelle des Math-Clients			
Nr.	**Ereignis**	**Auslöser**	**Antwort**
1C.	Benutzer gibt Daten ein	Dateneingabe	Matheaufgabe_mZ, Matheaufgabe_oZ
2C.	Ergebnis des Servers trifft ein	Ergebnis_oZ, Ergebnis_mZ oder Error	Bildschirmausgabe

1.2.1.2 Kontextdiagramm

Wie beim Punkt 1.2.1.1 Ereignistabelle gilt auch hier wieder, dass das Kontextdiagramm des Gesamtsystems bei normalem Betrieb dem des Math-Clients entspricht. Im Falle der Verwendung mit einem anderen Klienten muss jedoch auch das Kontextdiagramm des Math-Servers berücksichtigt werden. Hier würde der andere Klient an Stelle des Math-Client treten.

Diagramm 1.1



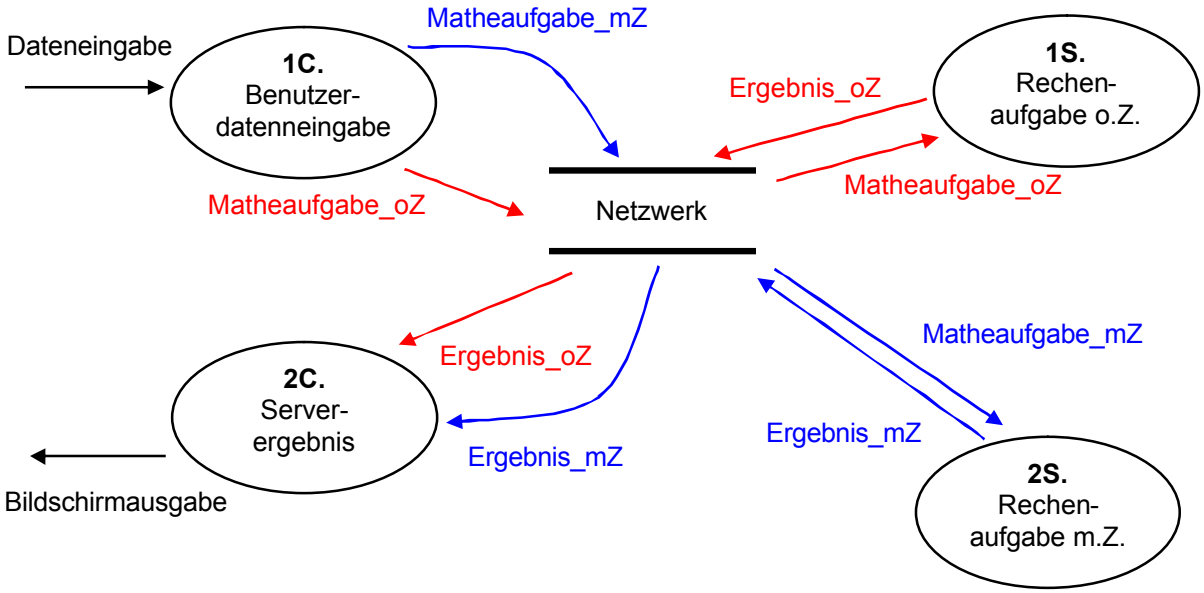
1.2.2 Verhaltensmodell

Die Applikation MathServer verwendet keine permanenten Speicher. Sämtliche Daten sind nur temporärer Natur und für die aktuellen Operationen notwendig. Aus diesem Grund stellen wir im Folgenden diese temporären Zwischenspeicher als Speicher dar. Ihre Strukturierung ist nicht durch harte Längen und Größenangaben definierbar, sie ist dynamisch.

1.2.2.1 Grobes Verhaltensmodell (vergröbertes primäres Verhaltensmodell)

Für das Verhaltensmodell verwenden wir die Ereignisnummern aus 1.2.1.1 Ereignistabelle und die farblichen Hervorhebungen des Kontextdiagramms 1.2.1.2. Die Datenströme übernehmen ebenfalls die Bezeichnungen aus der Ereignistabelle 1.2.1.1. Das Netzwerk, indem die Applikation betrieben wird, fungiert hier als Zwischenspeicher. Wurden zum Beispiel die Eingabedaten des Math-Clients abgeschickt, werden sie im Speicher freigegeben, sie entstehen erst auf der Serverseite durch den Empfang der Daten neu.

Diagramm 1.2



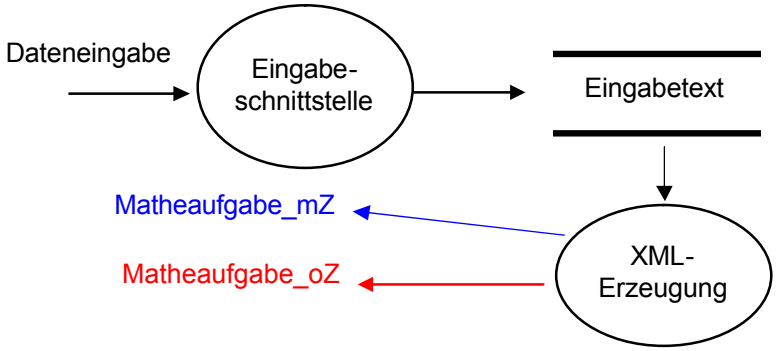
1.2.2.2 Primäres Verhaltensmodell

Bei dem Verhaltensmodell ist zu beachten, dass sowohl der Klient als auch der Server XML-Daten verarbeiten müssen. Dies wird prinzipiell nach dem gleichen Schema ablaufen, beide müssen sowohl XML-Daten zerlegen als auch erzeugen. Da beide Teile der Applikation denselben Prozess benutzen (aber auf verschiedenen Rechnern in verschiedenen Instanzen) taucht er sowohl bei den Diagrammen des Math-Client als auch des Math-Servers auf.

Im Folgenden werden Teilmodelle des Klienten mit „C“ und des Servers mit „S“ gekennzeichnet.

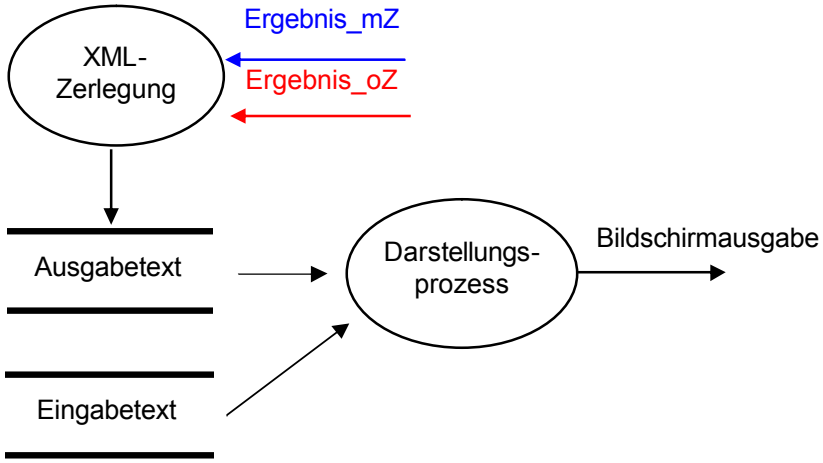
1.2.2.2.1C Teilmodell Benutzerdateneingabe

Diagramm 1.3



1.2.2.2.2C Teilmodell Serverergebnis

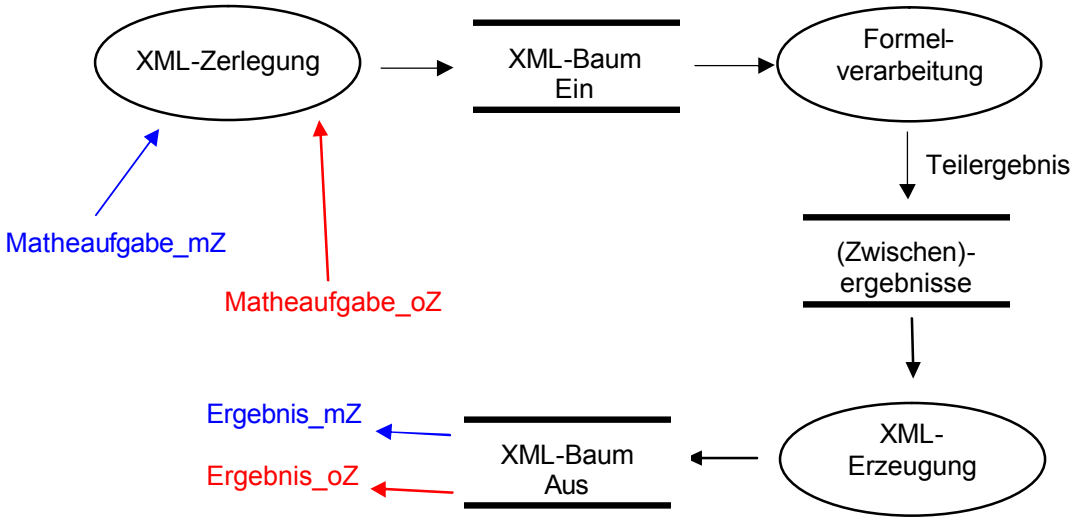
Diagramm 1.4



1.2.2.2.1S,2S,3S Teilmodelle ungültige_Daten, Rechenaufgabe_oZ, Rechenaufgabe_mZ

Die internen Abläufe innerhalb der Serveranwendung sind bei den Eingabefällen Rechenaufgaufgabe_oZ und Rechenaufgabe_mZ annähernd gleich. So können sich Eingabedaten sowohl bei der XML-Verarbeitung, der Formelverarbeitung als auch bei Berechnung der mathematischen Operationen als fehlerhaft herausstellen.

Diagramm 1.5

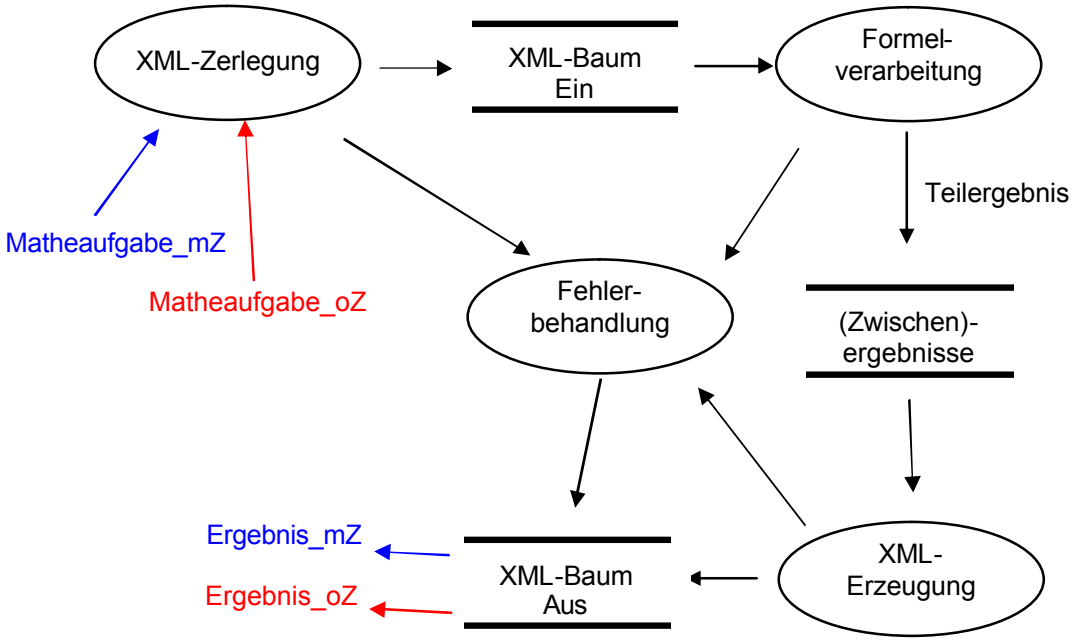


1.2.2.3 Verfeinern der Prozesse des primären Verhaltensmodells

1.2.2.3.S Verfeinerungen zu den Prozessen des Servers

Von jedem einzelnen Prozess können spezifische Fehler der Eingabedaten erkannt werden. Die restlichen Prozesse werden dann nicht ausgeführt.

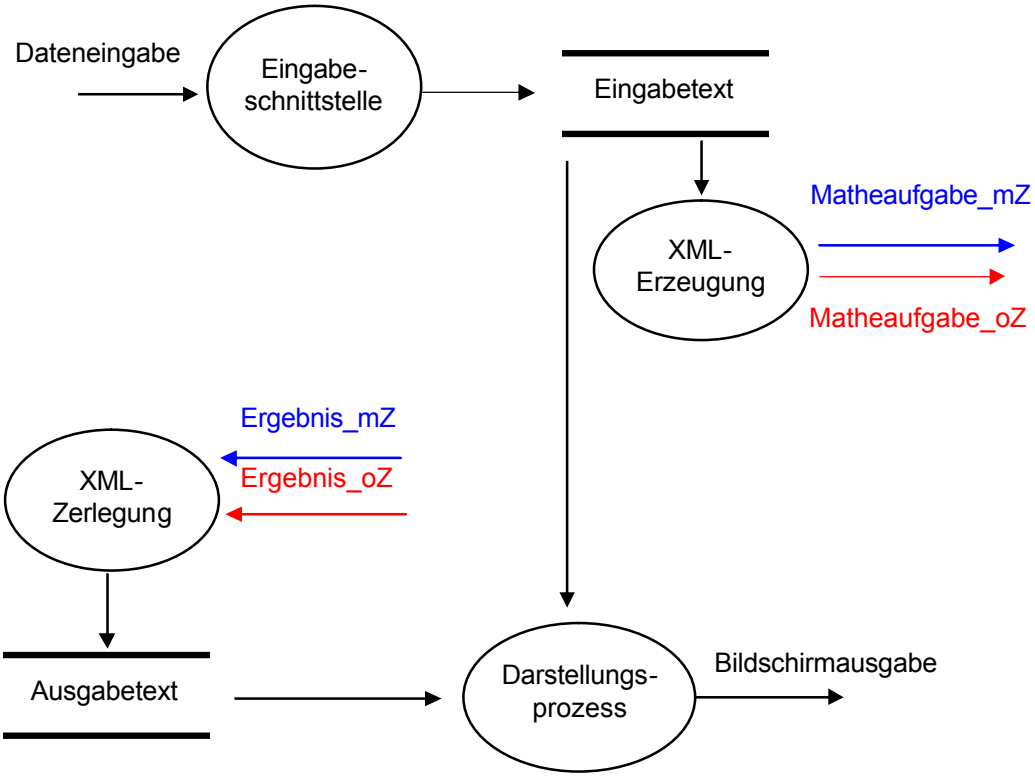
Diagramm 1.6



1.2.2.3.C Verfeinerungen zu den Prozessen des Klienten

Eine Fehlererkennung erlaubt das Abfangen von Falscheingaben und somit das Umgehen der Datenübertragung an den Server.

Diagramm 1.7



1.2.2.4 Datenkatalog

Wie gesagt ist die Struktur der temporären Speicher (im Weiteren wieder Speicher) der Applikation dynamisch und nicht durch feste Größen- oder Sequenzangaben definierbar. Sie lässt sich nur näherungsweise durch grammatikalische Konstrukte annähern, was im Folgenden im Datenkatalog und allen darauf beruhenden Betrachtungen versucht werden soll.

Der Datenkatalog enthält sowohl die Formate der Eingabedaten des Math-Servers und des Math-Clients. Verwended werden die allgemeinen Standards XML, MathML, HTTP und TCP/IP des W3C für die Ein- und Ausgabedaten.

Die Definition von XML, der Extensible Markup Language (XML) 1.0 (Second Edition), befindet sich im Internet unter http://www.w3.org/TR/REC-xml.

Die Definition von MathML, der Mathematical Markup Language (MathML) Version 2.0, befindet sich im Internet unter http://www.w3.org/TR/MathML2. Die zugehörige Grammatik kann über http://www.w3.org/tr/2003/wd-mathml2-20030411/appendixp.html erreicht werden.

Zu Datenübertragung verwenden wir ein HTTP-kompatibles Protokoll. Die Spezifikation zu HTTP (HyperText Transfer Protocol) befindet sich im Internet unter ftp://ftp.isi.edu/in-notes/rfc2616.txt.

Der eigentliche Datentransfer erfolgt über das Transfer Control Protocol / Internet Protocol (TCP/IP), welches ebenfalls weltweite Verbreitung genießt.

Grammatikalische Datendefinitionen des Katalogs werden in der Backus-Naur Form (BNF) notiert.

Der Klient besitzt zwei Ein- und Ausgabeinterfaces. Das Interface zum Netzwerk nutzt ebenfalls XML und MathML für die Ein- und Ausgabedaten. Das Interface zum Benutzer verwendet folgende Datendefinitionen für die Dateneingabe.

Die Definition der angeforderten Rechengenauigkeit findet sich unter 1.2.2.4.3. Aufgrund der begrenzten, vom Kunden zur Verfügung gestellten, Zeitressourcen, werden die Zahlen als Festkommazahlen implementiert, die eine feste Anzahl Vor- und Nachkommastellen besitzen.

1.2.2.4.1 Datendefinition der Eingabe-/Ausgabeschnittstelle des Klienten

Der Benutzer kann in die Eingabeschnittstelle einen Text entsprechend der folgenden Definition eingeben.

Mit Hilfe dieser Grammatik werden ebenfalls die unterstützten mathematischen Operation definiert, denn nur diese sind gültige Eingaben.

Im Grunde definiert die Grammatik einen beliebigen mathematischen Ausdruck, der auf den Operationen Addition, Subtraktion, Multiplikation, Division, Wurzel, Potenz, allgemeiner Logarithmus, Sinus, Kosinus, Tangens, Arcussinus, Arcuscosinus, Arcustangenz, natürlicher Logarithmus, Betrag und Fakultät basiert und die Konstanten Pi, Boltzmann-Konstante und Lichtgeschwindigkeit enthalten kann. Der Ausdruck kann beliebig geklammert werden.

Das Startsymbol ist <ausdruck>.

Tabelle 1.4

Element	Strukturbeschreibung
<konst>	{’pi’ | ’k’ | ’c’}
<ziffer>	{'0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'}
<dztrenn>	Benutzereinstellungsabhängig
<vzeich>	{'+' | '-' | ε}
<zfolge>	{<ziffer> | <ziffer> <zfolge>}
<zahl>	{ <vzeich> <zfolge> | <vzeich> <zfolge> <dztrenn> <zfolge> }
<objekt>	{<zahl> | <konst>}
<op>	{'+' | '-' | '*' | '/'}
<re>	{<objekt> <op> <objekt> | <ausdruck> <op> <objekt> | <objekt> <op> <ausdruck>, <ausdruck> <op> <ausdruck>}
<op2>	{’√’ | ’^’}
<powu>	{<objekt> <op2> <objekt> | <objekt> <op2> '(' <ausdruck> ')' | '(' <ausdruck> ')' <op2> <objekt> | '('<ausdruck>')' <op2> '(' <ausdruck> ')'}
<op3>	{’sin’ | ’cos’ | ’tan’ | ’arcsin’ | ’arccos’ | ’arctan’ | ’ln’}
<trig>	{<op3> <objekt> | <op3> '(' <ausdruck> ')'}
<log>	{<objekt> 'log' <objekt> | <objekt> 'log ' '(' <ausdruck> ')' | '(' <ausdruck> ')' 'log' <objekt> | '(' <ausdruck> ')' 'log’ '(' <ausdruck> ')' }
<fak>	{<zfolge> '!' | '('<ausdruck>')' '!'}
<betrag>	{'|' <objekt> '| '|' |' <ausdruck> '|'}
<ausdruck>	{<objekt> | <re> | <powu> | <trig> | <log> | <fak> | <betrag> }

Der Ausgabetext entspricht entweder einer Zeichenkette für die Fehlermeldung oder einer reellen Zahl in der Form <zahl> für End- und Zwischenergebnisse.

1.2.2.4.2 Liste der unterstützten MathML-Tags für den Server

Das Projekt unterstützt einen Subset des MathML-Standards, der für die Übertragung der gewünschten mathematischen Aufgaben ausreichend ist.

Dabei werden die Tags von MathML in drei Gruppen klassifiziert. Die erste Gruppe enthält Tags die unterstützt werden, dann folgen als Erweiterung geplante Tags und Tags deren Unterstützung in nächster Zukunft nicht vorgesehen ist.

Folgende Tags sind für die Formeldarstellung in den Datenströmen Matheaufgabe_oZ und Matheaufgabe_mZ zulässig.

Bei Serveranfrage und –antwort werden gewünschte Zwischenergebniss durch eindeutige id-Attribute identifiziert. z.B. <exp id="123" /> oder <apply id="abc"></apply>.

Tabelle 1.5.1

aktuell unterstützt	als Erweiterung geplant	noch nicht unterstützt
<abs />	<and/>	<annotation></annotation>
<apply></apply>	<arg/>	<annotation-xml></annotation-xml>
<arccos/>	<ceiling/>	<approx/>
<arcsin/>	<conjugate/>	<arccosh/> <arccot/> <arccoth/>
<arctan/>	<false/>	<arccsc/> <arccsch/> <arcsec/>
<cn></cn>	<floor/>	<arcsech/> <arcsinh/> <arctanh/>
<cos/>	<gcd/>	<bvar></bvar>
<divide/>	<imaginary/>	<card/>
<exp/>	<imaginaryi/>	<cartesianproduct/>
<exponentiale/>	<implies/>	<ci></ci>
<factorial/>	<inverse/>	<codomain/>
<infinity/>	<lcm/>	<complexes/>
<ln/>	<not/>	<compose/>
<log/>	<or/>	<condition></condition>
<minus/>	<real/>	<csymbol></csymbol>
<pi/>	<true/>	<curl/>
<plus/>	<xor/>	<declare></declare>
<power/>	<and/>	<degree></degree>
<quotient/>	<arg/>	<determinant/>
<rem/>		<diff/>
<root/>		<divergence/>
<sep/>		<domain/>
<sin/>		<domainofapplication>
<tan/>		<emptyset/>
<times/>		<eq/>

Tabelle 1.5. 2

noch nicht unterstützt	noch nicht unterstützt	noch nicht unterstützt
<equivalent/>	<lt/>	<rationals/>
<eulergamma/>	<matrix></matrix>	<reals/>
<exists/>	<matrixrow></matrixrow>	<reln></reln>
<factorof/>	<max/>, <min/>	<scalarproduct/>
<fn></fn>	<mean/>, <median/>	<sdev/>
<forall/>	<mode/>	<sec/> <csc/> <cot/>
<geq/>	<moment/>	<sech/> <csch/> <coth/>
<grad/>	<momentabout>	<selector/>
<gt/>	<naturalnumbers/>	<semantics></semantics>
<ident/>	<neq/>	<set></set>, <setdiff/>
<image/>	<notanumber/>	<sinh/> <cosh/> <tanh/>
<in/>	<notin/>, <notprsubset/>	<subset/>
<int/>	<notsubset/>	<sum/>
<integers/>	<otherwise></otherwise>	<tendsto/>
<intersect/>	<outerproduct/>	<transpose/>
<interval></interval>	<partialdiff/>	<union/>
<lambda></lambda>	<piece></piece>	<uplimit></uplimit>
<laplacian/>	<piecewise></piecewise>	<variance/>
<leq/>, <limit/>	<primes/>	<vector></vector>
<list></list>	<product/>	<vectorproduct/>
<lowlimit></lowlimit>	<prsubset/>	

1.2.2.4.3 Grammatik für die Datenströme zwischen Math-Client und Math-Server

Wie bereits erwähnt wird ein HTTP-kompatibles Protokoll verwendet. Da es nur der Datenübertragung dient, beschränken wir uns darauf, den Befehl „GET“ zu unterstützen. Anstelle des URL-Teils wird das Wort „solution“ eingetragen. Die Antwort des Servers wird entweder den Kode „200 OK“ oder einen von definierten Fehlerkode „8xx Fehler“ enthalten.
Hierbei kann die Anzahl der gewünschten Vorkommastellen (decimals) und Nachkommastellen (fractals) definiert werden.
Das Startsymbol für den gesamten Traffic ist <traffic>, für die Datenübertragung Klient-Server <anfrage> und Server-Klient <antwort>.

Tabelle 1.6

Element	Strukturbeschreibung
<http-befehl>	{"GET"}
<typ>	{"solution decimals=" <zfolge> “ fractals=“ <zfolge> }
<LF>	Zeilenumbruch
<leerzeile>	<LF><LF>
<xml-daten>	XML-Text für Anfrage- und Antwortdaten
<ergebnis-nr>	{"200 OK" | "8" <ziffer><ziffer> " Fehler"}
<anfrage>	{<http-befehl>" "<typ>" HTTP/" <ziffer> "." <ziffer> <leerzeile> <xml-daten>}
<antwort>	{"HTTP/" <ziffer> "." <ziffer> " " <ergebnis-nr> <LF> “Content-Type: text/mathml+xml“ <leerzeile> <xml-answer>}
<traffic>	{<anfrage> | <antwort>}

1.2.3.4.4 Struktur der XML-Daten der Serverantwort

Die XML-Daten der Antwort (entspricht dem Symbol <xml-daten> der <antwort> im Punkt 1.2.2.4.3 Tabelle 1.6) des Servers erfolgt nach MathML-Syntax.
Die einzelnen Ergebnisse (Endergebnis und Zwischenergebnisse) werden je in <cn>-tags (<cn> </cn>) geklammert.
Der <cn>-tag, als einiger zulässiger Datenübergabe-Tag, erfährt eine id-Nummer als Attribut <cn id = [ganze positive Zahl]>.
Die id-nummer stellt die Verknüpfung zwischen dem jeweiligen Ergebnis und dem zugehörigen Berechnungsabschnitt dar.

Tabelle 1.7

Element	Strukturbeschreibung
<xml-pi>	XML Processing-Instructions
<xml-dt>	XML Comment
<xml-mathml>	XML Comment
<xml-math>	“<math>“ <xml-cnc> “</math>“
<xml-cn>	“<cn id=’“ <zfolge> “’>“ <zahl> “</cn>“
<xml-cnc>	<xml-cn> | <xml-cn> <xml-cnc>
<xml-answer>	<xml-pi> <xml-dt> <xml-mathml>

Beispiel 1.1

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- DOCTYPE math PUBLIC "-//W3C//DTD MathML 2.0//EN" "dtd/mathml2.dtd" -->
<!—math xmlns="http://www.w3.org/1998/Math/MathML" -->
<math>
<cn id=“0“>587459236482364972407239462845185.234236487235648235423453453</cn>
<cn id=“1“>122344482364972407239462845185.345896472323356482354234534567455563</cn>
<cn id=“2“>0.45345673457347768395634564569784568745645896745896734523334434555 6</cn>
<cn id=“3“>4.45723498623478263487233094782394672183210742893864923674923364</cn>
<cn id=“4“>234572374.34926347236462329639723972937892374923749273498273497346 4</cn>
</math>
```

1.2.3.4.5 Struktur der Zwischenergebnisse

Die Teilergebnisse werden in der Form <xml-cn> übergeben, die 1.2.3.4.4. definiert wurde, und im Speicher Zwischenergebnisse als Feld solcher Daten gespeichert.

1.2.2.5 Struktur der Speicher (grafische Ergänzung zu Inhalten des Datenkatalogs)

1.2.2.5.1 Struktur der Eingabeschnittstelledaten des Klienten

Diagramm 1.6.1

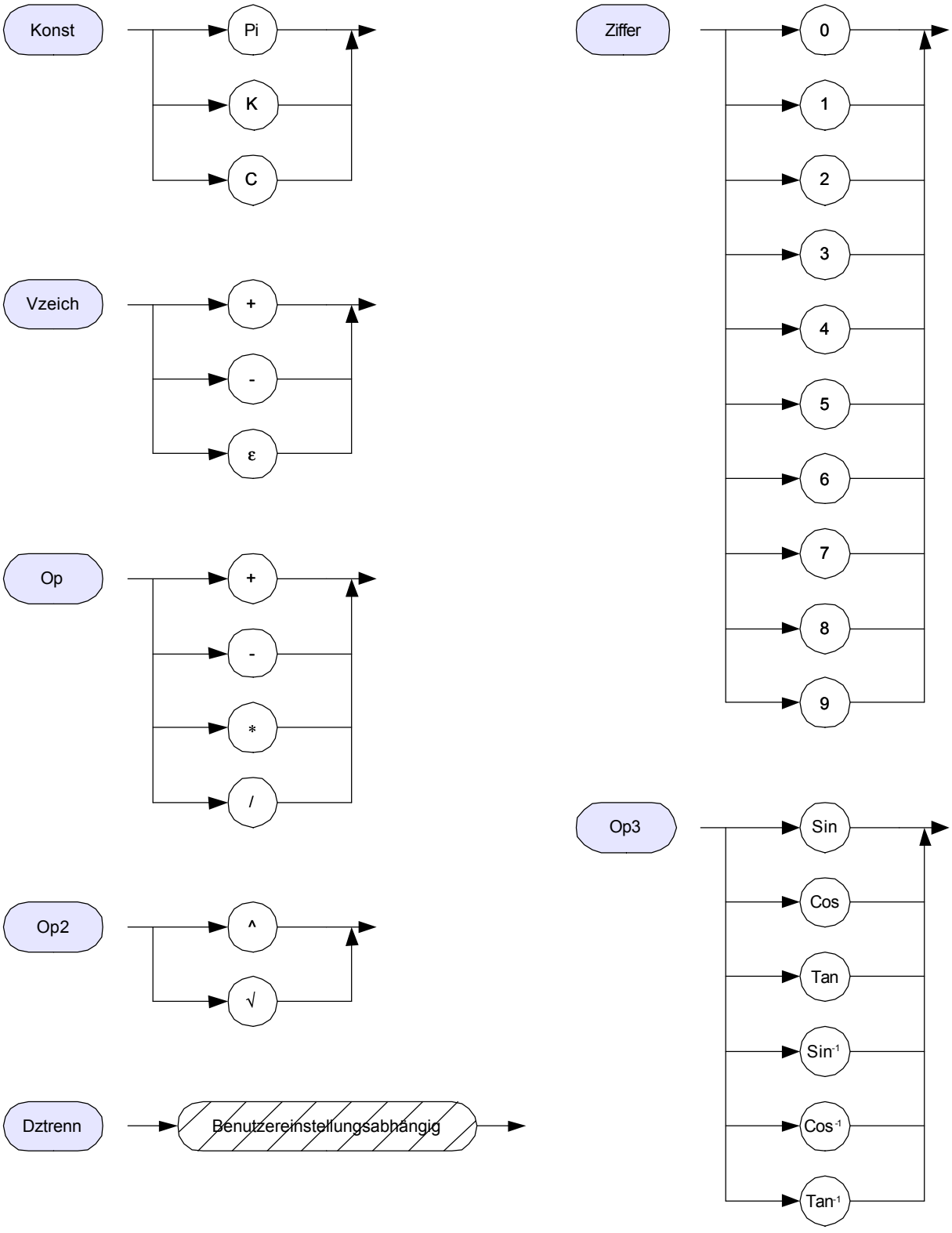


Diagramm 1.6.2

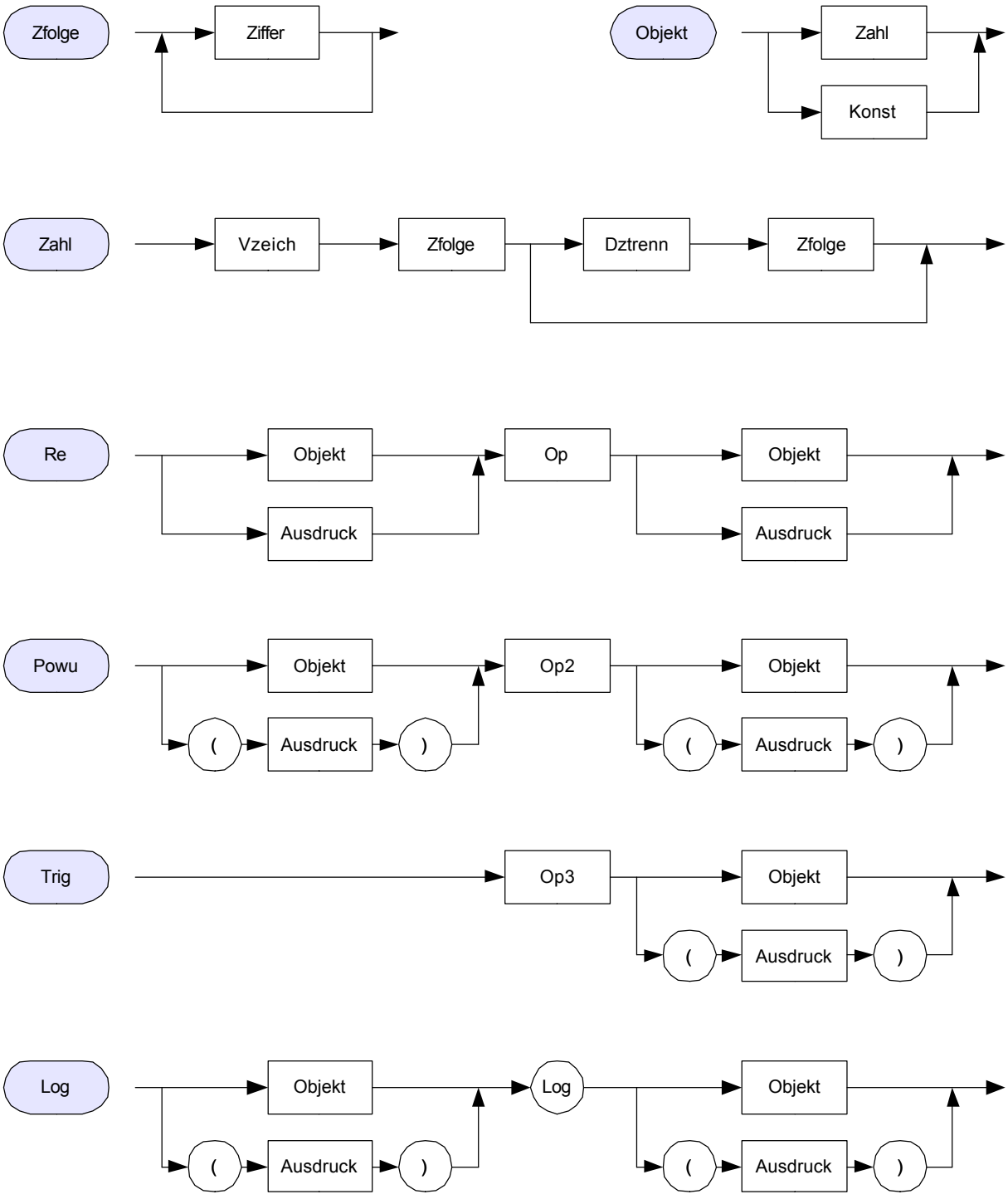
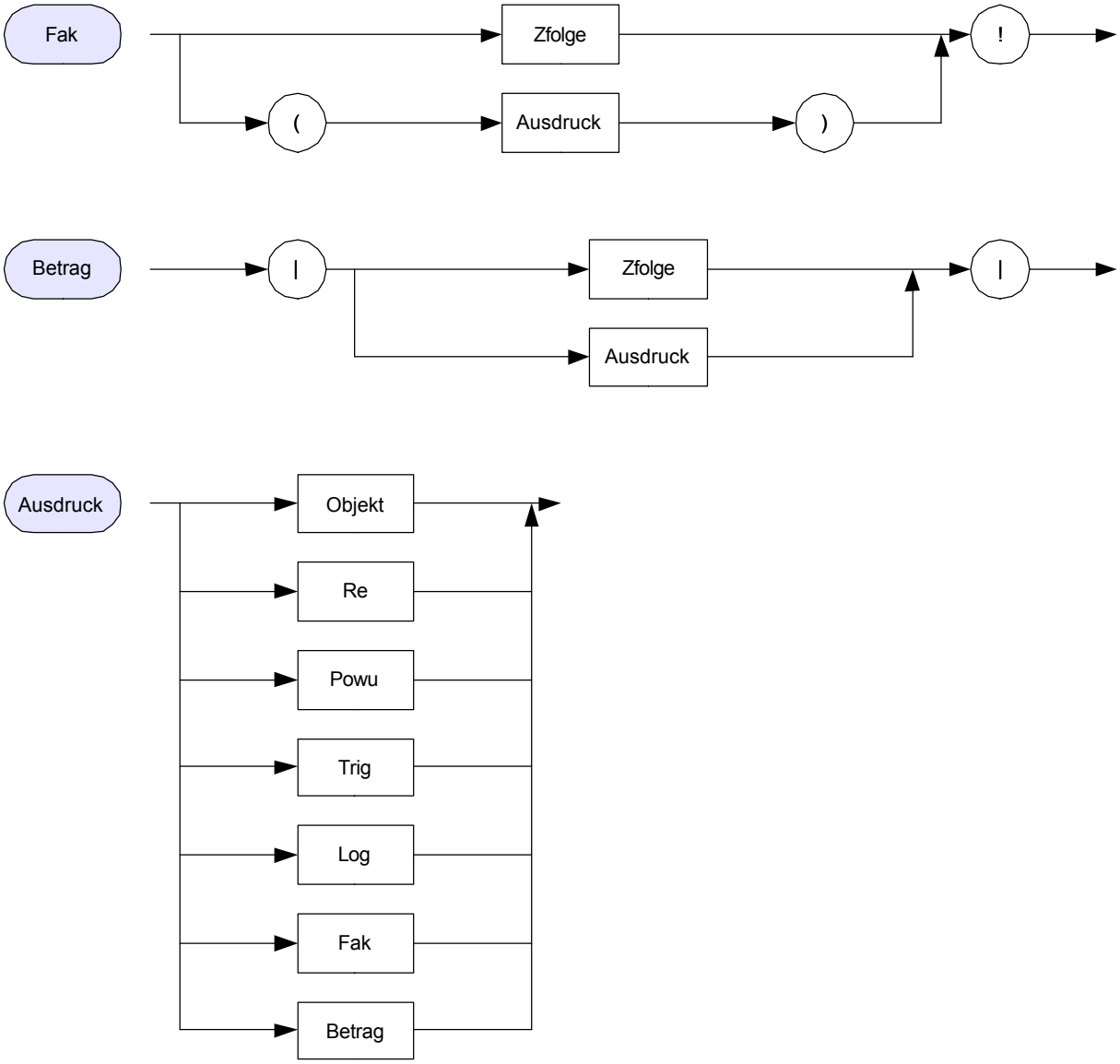


Diagramm 1.6.3



1.2.2.5.2 Struktur des Netzwerktraffics

Um die Übersicht zu verbessern, wurden einzufügende Leerzeichen durch den Unterstrich („_“) ersetzt.

Diagramm 1.7.1

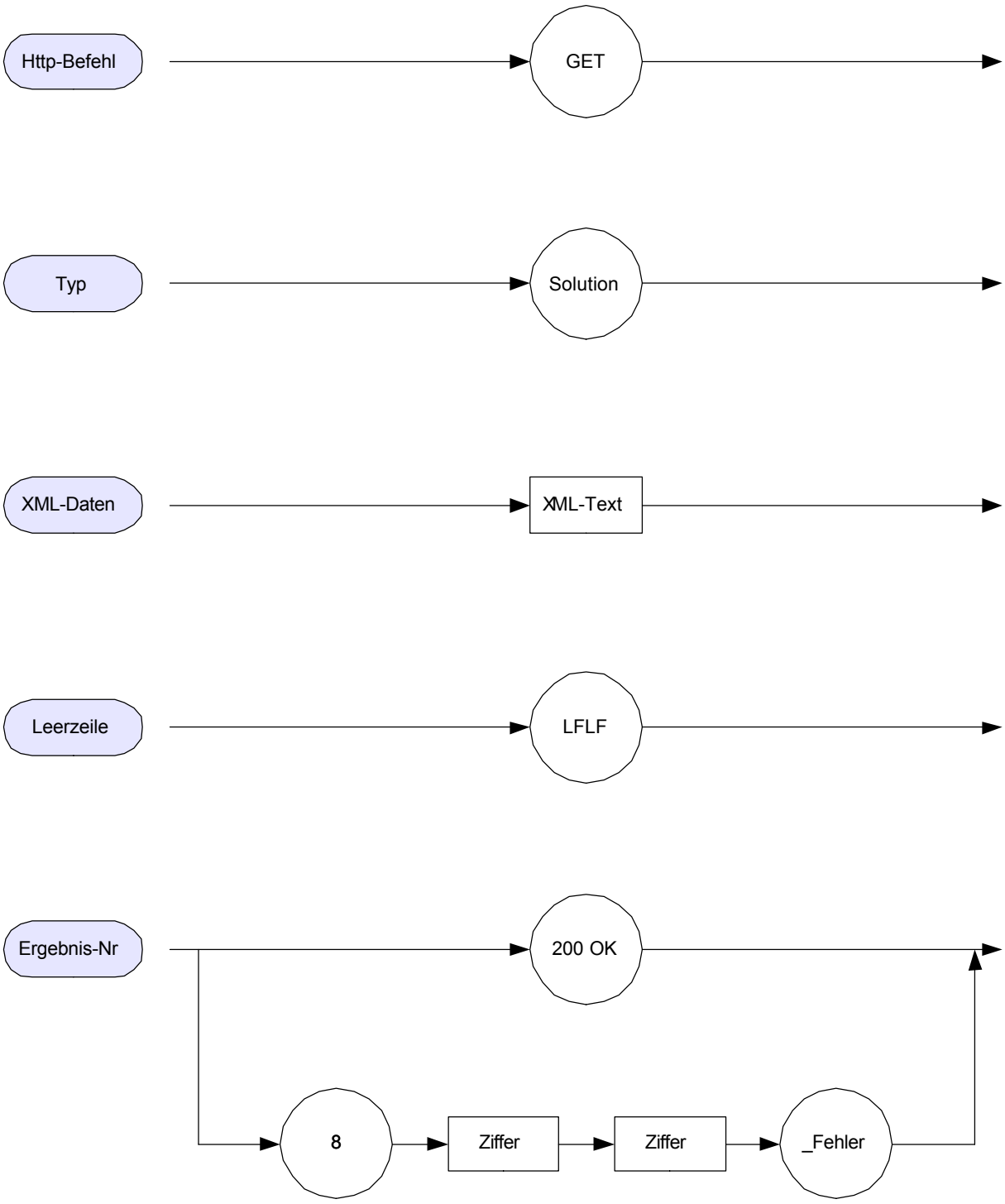
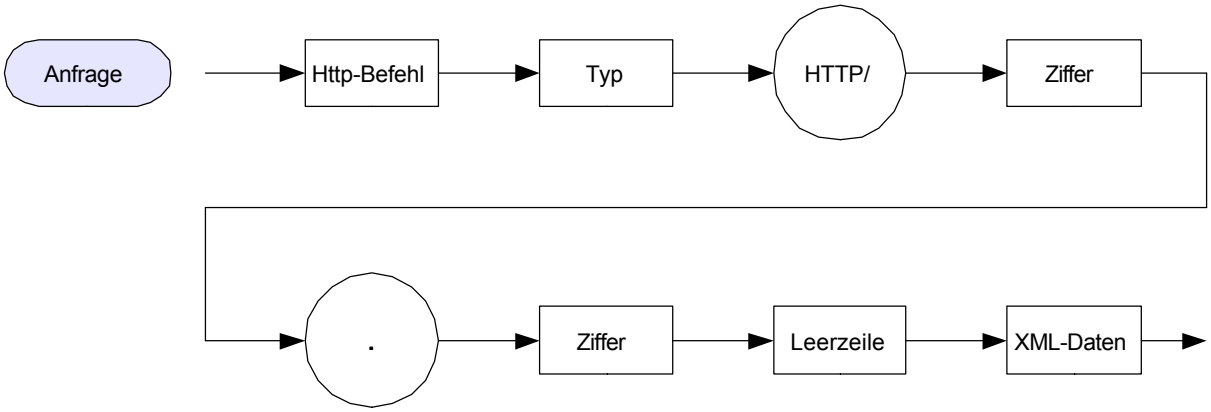
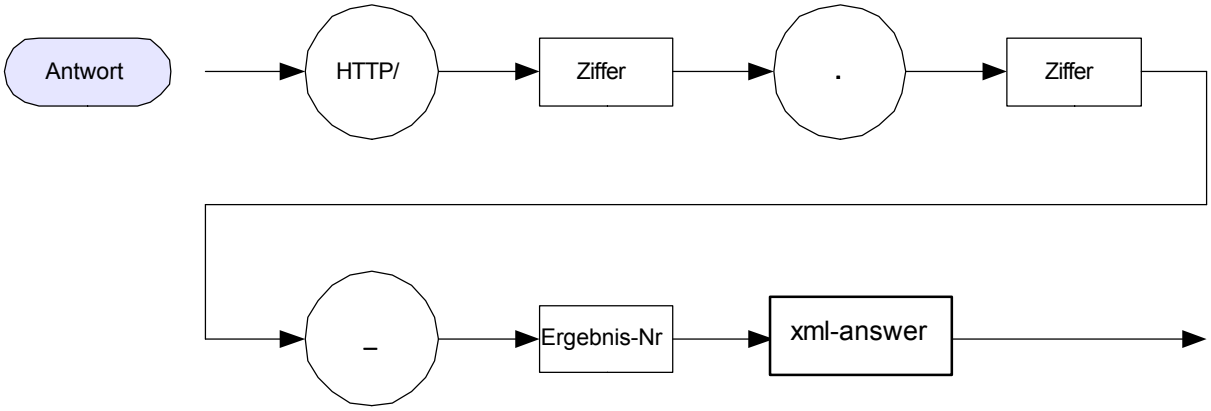
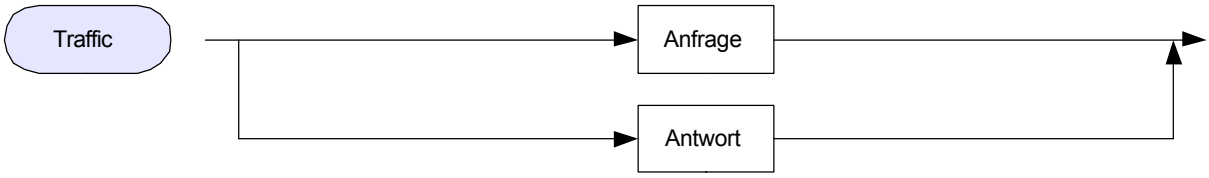


Diagramm 1.7.2





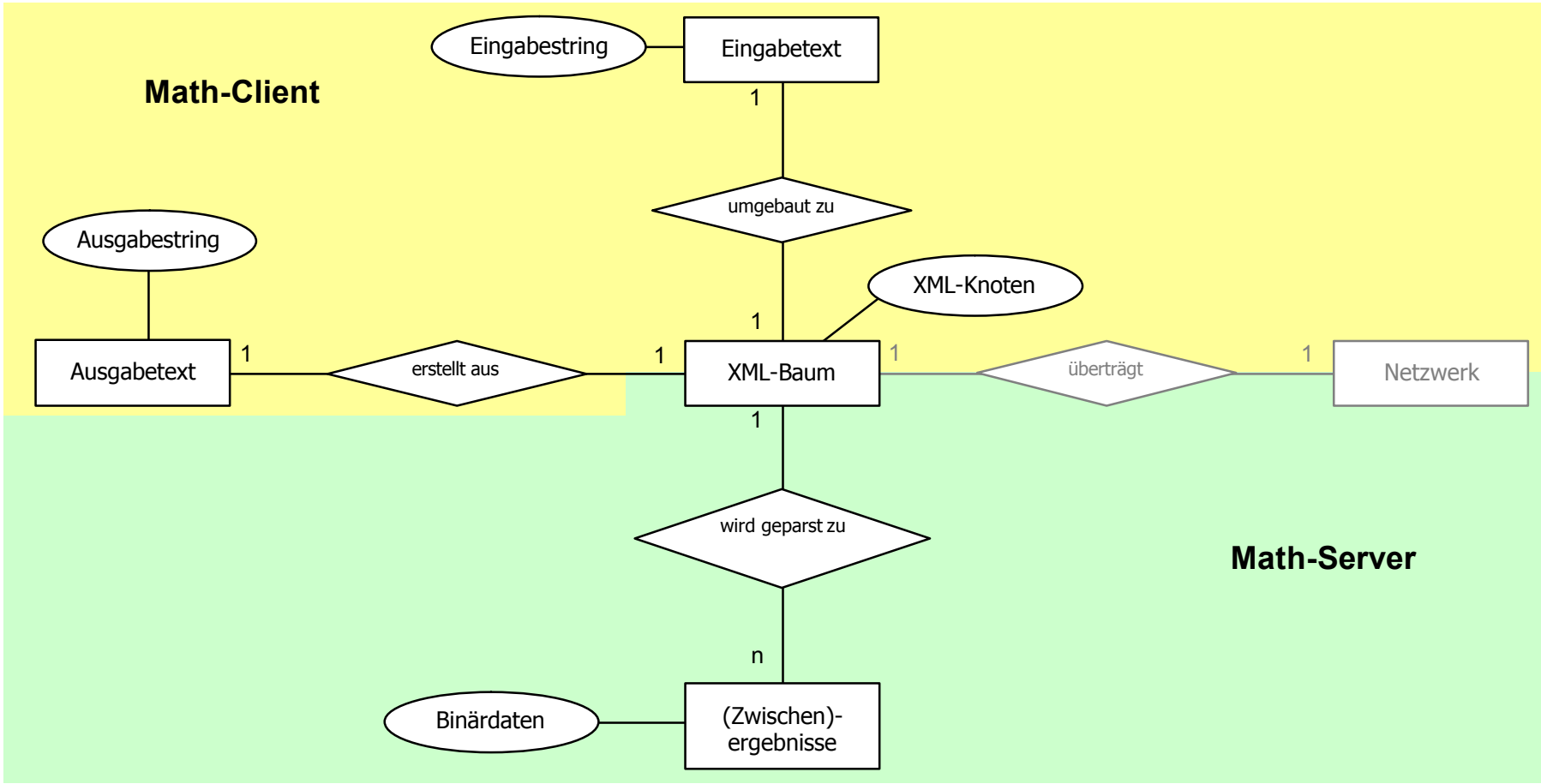


1.2.2.6 Beziehungen zwischen den Speichern (ERD)

Da wir keine persistenten Speicher verwenden, macht die Angabe eines ERD eigentlich auch keinen Sinn. Um jedoch wenigstens etwas ähnliches anzufertigen um diese Art des Softwareengineering zu üben, wurde das folgende Diagramm eingebracht.

Wie bereits das Ereignisdiagramm 1.2.1.1 als auch als auch das Kontextmodell 1.2.1.2 wird auch das Entity Relationship Diagramm in Klienten- und Serverteil untergliedert, wobei beide die Datenstrukturen der XML-Baum verwenden. Wieder wird das Netzwerk als Pseudo-Speicher dargestellt.

Diagramm 1.7



1.2.2.7 Prozessspezifikation

1.2.2.7.1 Eingabeschnittstelle

Prozess: eingabeschnittstelle	Datum:	04.05.2003
	Bearbeiter:	Rico Roßberg

Voraussetzungen: der Nutzer hat einen <ausdruck> A eingegeben

```
Begin
 Erzeuge Wurzelknoten R
 NeuenKnotenAnfügen(A, R)
End

Unterprozess NeuenKnotenAnfügen (Eingabeausdruck, Vaterknoten)
Begin
 Erzeuge neuen XML-Knoten k
 If Eingabeausdruck enthält <op> oder <op2> then
  Begin
   k.daten:=gefundener <op> oder <op2>
   For each übrige Teilausdrücke NeuenKnotenAnfügen(Teilausdruck, k)
  end
 Else
  k.daten:=eingabeausdruck
 End

 Knoten k an Vaterknoten anfügen
end
```

1.2.2.7.2 XML-Erzeugung

Prozess: xml_erzeugung	Datum:	02.05.2003
	Bearbeiter:	René Kreiner
		Roland Fischer

Voraussetzungen: B ist ein gültiger XML-Baum

```
begin
ausgabetext = x.anfangstag + x.daten + x.schlusstag
end

objektorientierter unterprozess x.daten
begin
zeichenkette s = gespeicherte datenzeichenkette
for each a in unterknoten do
  begin
  zeichenkette l = a.anfangstag + a.daten + a.schlussdaten
  füge l in s an gespeicherter position ein
  end
gib s zurück
end
```

1.2.2.7.3 XML-Zerlegung

Prozess: xml_zerlegung	Datum:	02.05.2003
	Bearbeiter:	René Kreiner
		Roland Fischer

Voraussetzungen: Zu untersuchende Zeichenkette Z ist nicht leer

```
begin
erstelle dokumentknoten
knotenerkennung (dokumentknoten, Z)
if Fehler then lösche dokumentknoten
end

unterprozess knotenerkennung (vaterknoten, zeichenkette Z)
begin
while Z nicht leer und kein Fehler do
  for each x in bekannte Knotenarten do
        begin
        if x erkennt Z then
          begin
          erstelle einen neuen Knoten k vom typ x
          finde Ende von x in Z
          teile diesen ganzen Teil ab und speichere ihn in k
          hänge ihn an vaterknoten an
          if x kann Unterknoten enthalten then
            begin
            knotenerkennung (k, Z)
            end
          end
        end
  if keine Knotenart passt auf z then Fehler
  end
end
```

1.2.2.7.4 Darstellungsprozess

Prozess: darstellungsprozess	Datum:	18.05.2003
	Bearbeiter:	Rico Roßberg

Voraussetzungen: Antwort vom Server wurde erhalten, in Ausgabefenster wurde Eingabetext grafisch dargestellt, Nutzer hat mit Schaltflächen „Nächstes/Vorheriges Teilergebnis“ einen Formelteil ausgewählt

```
Begin
 If Serverantwort enthält Fehlermeldung then
  Zeige Fehlermeldung an
 Else
  Begin
   Durchsuche Eingabetext nach dem vom Nutzer ausgewählten Formelteil
   Ermittle ID-Attribut des gefundenen Formelteils
   Durchsuche Ausgabetext nach dieser ID
   Zeige dieser ID zugeordnete Daten als Ergebnis an
  End
 end
end
```

1.2.2.7.5 Formelverarbeitung

Prozess: formel_verarbeitung	Datum:	18.05.2003
	Bearbeiter:	Rico Roßberg
		Thomas Ziegs

Voraussetzungen: Prozess erhält Eingabe-XML-Knoten E und gibt Ausgabe-XML-Knoten A zurück.

```
Begin
 Für alle Unterknoten x des XML-Baums E
  Begin
   Berechne Teilergebnisse von x                ; durch Selbstaufruf
     mit Hilfe mathematischer Funktionen
   und merke diese
  Berechne eigenes Ergebnis aus Daten von E und untergeordneten Teilergebnissen von E
   mit Hilfe mathematischer Funktionen
  Erzeuge Knoten A und setze Daten von A auf Ergebnis von E
 End
End
```

1.2.2.7.6 Fehlerbehandlung

Prozess: fehlerbehandlung	Datum:	02.05.2003
	Bearbeiter:	Thomas Weise

Voraussetzungen: Fehler werden mit den COM-/OLE-spezifischen Fehlerkodes idendifiziert.

```
begin
finde Fehler in Fehlertabelle
if Fehler gefunden then
  begin
  gib Fehlertext aus
  end
else
  begin
  gib Fehlermeldung aus, dass unbekannter Fehler aufgetreten ist
  end
end
```

1.3 Definition der Nutzerschnittstelle

Das System besitzt nur ein Fenster, welches jedoch über mehrere Registerkarten verfügt. Das Fenster, die Registerkarten und alle Steuerelemente haben fest vorgegebene, nicht vom Benutzer änderbare Größen. Die hier gezeigten Screenshots sind in Originalgröße dargestellt.

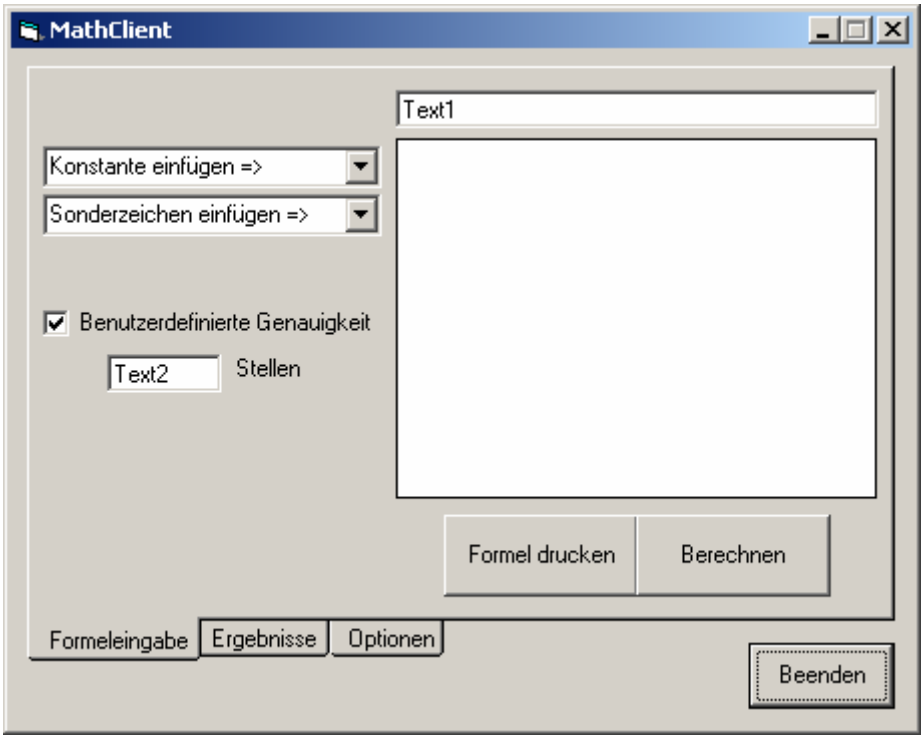
1.3.1 Ein- und Ausgabegeräte

Vorgesehene Eingabegeräte sind Tastatur und Maus, die Ausgabe erfolgt über den Bildschirm, wobei bestimmte Daten aber auch gedruckt werden können.

1.3.2 Layoutentwurf

Alle Bedienelemente verwenden die auf dem jeweiligen PC eingestellten Systemfarben.

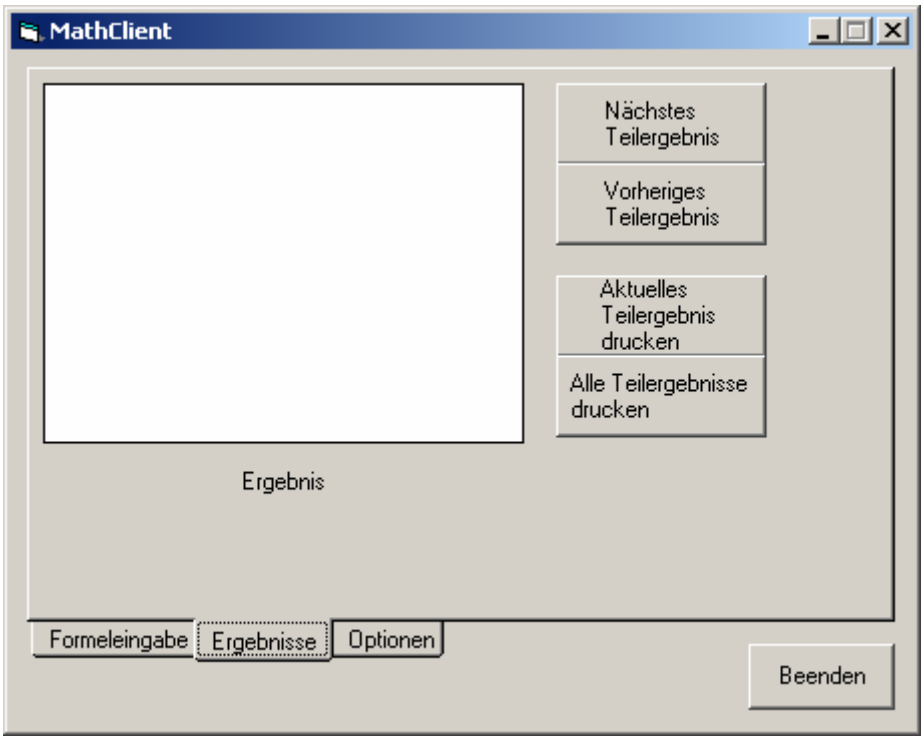
1.3.2.1. Registerkarte Formeleingabe



In der PictureBox „Eingabetext“ wird die bisher eingegebene Formel grafisch dargestellt, es kann jedoch nicht direkt Text eingegeben werden; über die Tastatur eingebbare Zeichen werden in die Textbox „Text1“ eingegeben, andere Zeichen und Konstanten über die Dropdown-Listen „Konstante einfügen“ und „Sonderzeichen einfügen“.

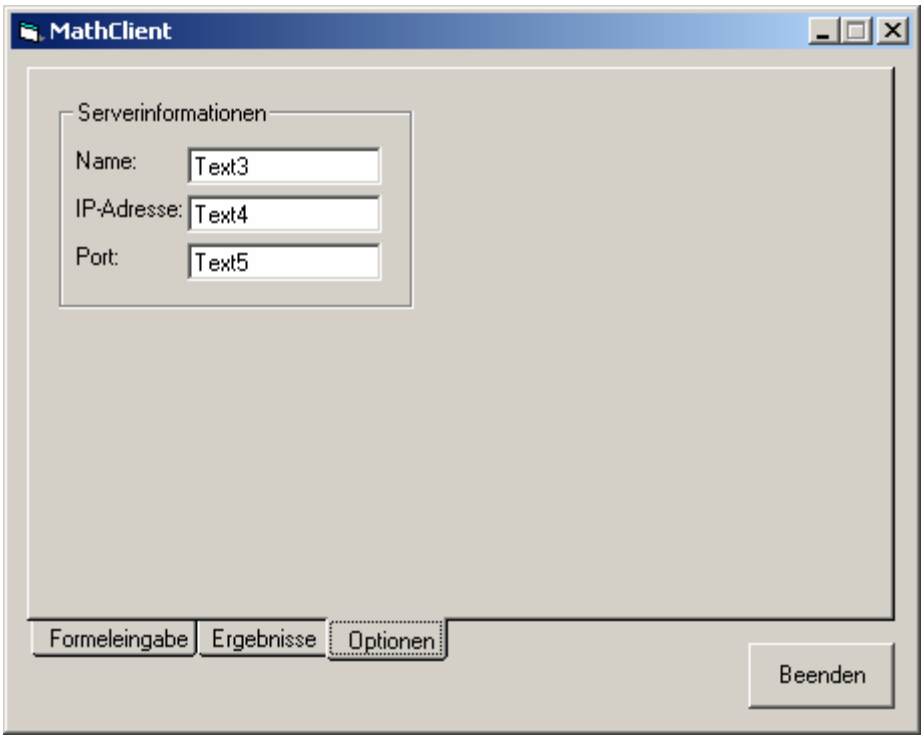
In der Textbox „Text2“ kann die gewünschte Rechengenauigkeit angegeben werden, falls die Checkbox „Benutzerdefinierte Genauigkeit“ aktiviert wird.
Mit der Schaltfläche „Formel drucken“ wird die in „Eingabetext“ dargestellte Formel ausgedruckt, die Schaltfläche „Berechnen“ löst den Berechnungsvorgang aus und wechselt zur Registerkarte Ergebnisse.

1.3.2.2. Registerkarte Ergebnisse



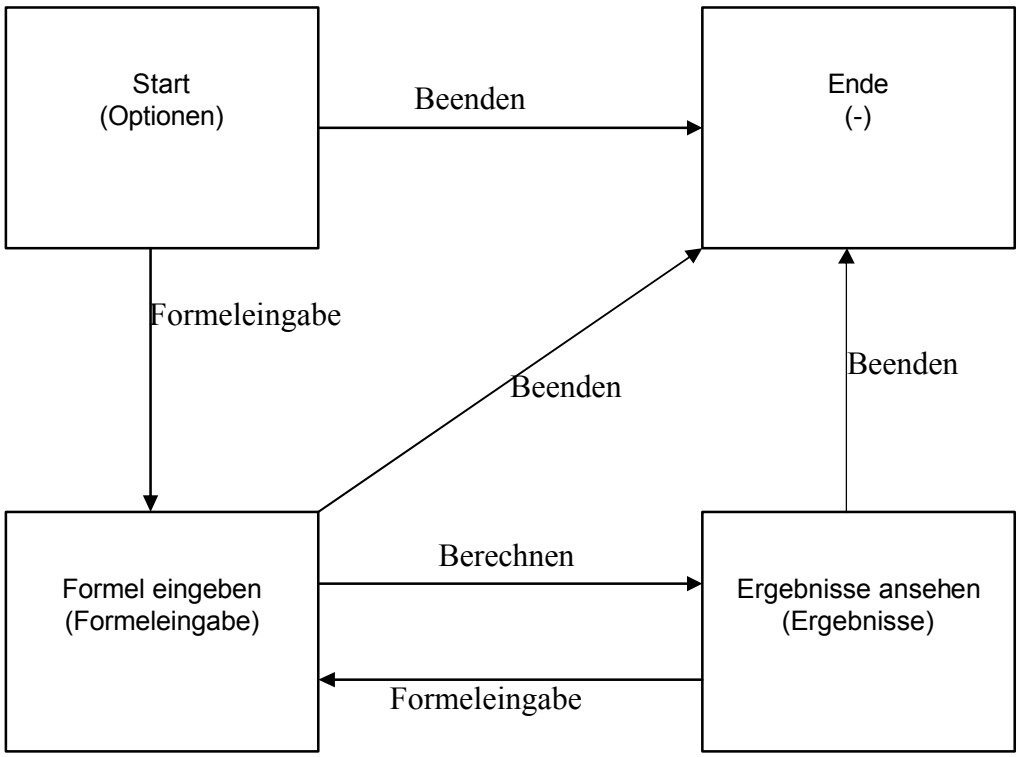
In der PictureBox „Ausgabetext“ wird wiederum die eingegebene Formel dargestellt, wobei aber diesmal jeweils die ganze Formel oder nur ein Teil der Formel durch Markierung hervorgehoben wird. Das Ergebnis des markierten Teils wird im Textfeld „Ergebnis“ angezeigt. Mit den Schaltflächen „Nächstes Teilergebnis“ und „Vorheriges Teilergebnis“ wird durch alle möglichen Teilergebnisse durchgeschalten.
Mit Hilfe der Schaltflächen „Aktuelles Teilergebnis drucken“ und „Alle Teilergebnisse drucken“ können jeweils auch Teilergebnisse ausgedruckt werden.

1.3.2.3. Registerkarte Optionen



In die Textboxen „Text3“, „Text4“ und „Text5“ werden Informationen, die zur Kommunikation mit dem Serverteil des Systems notwendig sind, eingetragen.

1.3.3 Zustandsdiagramm



2 Spezifikation der operationellen Anforderungen

2.1 Operationelle Anforderungen an die Daten und die Datenbasen

Für die Abschätzung des Speicherbedarfs der beiden Programme der Applikation MathServer werden folgende vereinfachende Annahmenden getroffen:

Zeichenketten im MathServer sind dynamisch, ihre Länge ist nahezu beliebig. Sie werden als Objekte dargestellt. Die Objekte beinhalten je ca. 30 Bytes an Steuerinformation. Sie allokieren den Speicher für die eigentliche Zeichenkette. Eine Zeichenkette im internen Format besteht aus Unicode-Zeichen, das heißt jedes Zeichen belegt zwei Bytes. Eine Zeichenkette mit n Zeichen belegt also 30 + 2 * n Bytes. Da sie aus maximal 2^{31} Zeichen bestehen kann, würde dies einen maximalen Speicherbedarf von 2 * 2^{31} + 30 Bytes (= 4294967326 Byte) bedeuten. Die 32 bit Betriebssysteme (siehe 4.1.2 Betriebssysteme) beschränken die Speichervergabe an ein Programm jedoch auf maximal 4 Gigabyte. Eine Zeichenkette minimaler Länge (0) würde 30 Byte Speicher belegen.

Die XML-Daten und Daten der verschiedenen Baumstrukturen der Parser (XML-Verarbeitung, Formelberechnung) werden durch Zeichenketten beschrieben.

Auch die reellen Zahlen (CReell) in den einzelnen Berechnungen werden durch Objekte (ca. 50 Byte Steuerinformation) dargestellt. Für n Dezimalstellen werden weitere $\left\lceil \frac{\log_2 10^n + 1}{32} \right\rceil * 4$ Bytes benötigt. Man könnte also maximal $\left(\log_{10}\left(2^{2^{32}} - 1\right)\right) + 1$ Dezimalstellen einer reellen Zahl darstellen, was eine zu große Zahl ist, um sie mit einem Standardtaschenrechner zu berechnen. Allgemein kann man von im Durchschnitt von nicht mehr als 1000 Dezimalstellen ausgehen, was einem Speicherbedarf von 416 + 50 = 466 Bytes entspricht.
Dadurch jedoch, dass die reellen Zahlen mit Zeichenketten vom und zum Klienten übergeben werden, die (auf Grund der obigen Überlegungen) auf 2^{31} Zeichen beschränkt sind, ist es nicht möglich, Zahlen mit mehr als 2^{31} (ca. 2 Milliarden) Stellen darzustellen. Dies entspricht einem Speicherbedarf etwa 851 MB.

Auch Baumstrukturen werden durch Objekte dargestellt (Knotenobjekt), wobei jeder Knoten (bzw. Blatt) einer Instanz entspricht. Solche Objekte besitzen im Durchschnitt 30 bis 70 Bytes an Steuerinformation. Knoten die Unterknoten enthalten können, erfordern dass die ihnen zugeordneten Objekte eine Liste mit den ihnen untergeordneten Knoten und zugeordneten Objekten besitzen. Eine solche Liste benötigt für n Unterobjekte 4 * n Bytes Speicher. Unterknoten können gegebenenfalls wieder Unterknoten enthalten.

Sämtliche numerischen Standardzahlen (also solche, die nicht in den Kalkulationen verwendet werden, sondern zum Beispiel die Länge einer Zeichenkette speichern) sind auf 4 Bytes Speicherbedarf normiert.

Die Struktur der Elemente entspricht der Struktur in 1.2.2.4 Datenkatalog.

Wie bereits in Punkt 1.2.2 erklärt, werden keinerlei permanente Speicher verwendet. Sämtliche Daten und Datengrößen sind dynamisch und besitzen keine festen Größen (bis auf die auf der vorigen Seite erwähnten Steuerinformationen). In diesem Sinne ist es nicht möglich, eine dem normalen Standard entsprechende Anforderungstabelle aufzustellen. Wir beschränken uns also an dieser Stelle darauf, eine grobe Abschätzung basierend auf angenommen Entwicklungsrichtlinien anzunehmen. Die exakten Definitionen sind dann implementationsabhängig und können hier noch nicht bestimmt werden.

Dafür wird es möglich weitaus umfangreichere Abschätzungen (2.1.3), Qualitätsanforderungen (3.) und Anforderungen an die Umgebung (4.) zu stellen.

Für die Datenkalkulation vereinfachen wir den XML-Baum ganz enorm.

Tabelle 2.1

Element	**Struktur**	**Bytes min**	**max**
<string n>	Zeichenkette der Länge n (CString)	2 * n + 30	
<tag n>	<string n>	2 * n + 30	
<xmlknoten>	<tag a> <string b> <tag a+1> | <tag a> | <tag a> <string b> <xmlknoten> <tag a+1>	2 * (2*a+b+1) + 90	k * min
<xml-baum>	<node>	2 * (2*a+b+1) + 90	k * min
<reelle_Zahl>	<CReell>	$\left\lceil \frac{\log_2 10^{\text{Stellenzahl}} + 1}{32} \right\rceil * 4 + 50 = \wp$	
<eingabetext>	<string n>	2 * n + 30	
<ausgabetext>	<string n>	2 * n + 30	
<netzwerktraffic>	<string a> <string b>	2 * (a + b + 30)	

Man erkennt, dass die grundlegenden Datentypen stets von Zeichenketten und den binären Daten der reellen Zahlen abgeleitet werden. Prinzipiell werden diese lediglich um Steuerinformationen erweitert. Anders als bei den Tabellen in der Vorlage kann hier zwischen minimaler und maximaler Größe unterschieden werden, da sich diese dynamisch errechnen.

2.1.1 Speicher (Angaben zum Speicherbedarf in Byte)

Aufgrund unserer Erfahrung mit der Programmierung wird es uns möglich, den zu erwartenden Umfang an Speichern abzuleiten. Wir beziehen unsere Abschätzungen auf die in Tabelle 2.1 entwickelten Formeln und Abschätzungen des Traffics der Multi-Thread Client-Server Applikation. Dabei gehen wir von normaler Benutzung bis kritischer Belastung aus.

2.1.1.C Speicher des MathClient

Tabelle 2.2

Speicher	Struktur	Datentyp	Größe	Minimum	Mittel	Maximum
Eingabetext	<ausdruck>	Zeichenkette (CString)	(Länge * 2) + 30	30	2 kB	4 GB
Ausgabetext	<zahl>	Zeichenkette (CString)	(Länge * 2) + 30	30	4 kB	4GB
XML-Baum	Implementationsabhängig	Knotenobjekt mit Unterknoten (CObjectList)	Anzahl Knoten * 70 + (Anzahl Knoten – 1) * 4	70	50 kB	4 GB
Der **Gesamtbedarf** kann auf Grund der gewählten Betriebssysteme niemals 4 GB überschreiten. Bei den Werten handelt es sich wegen Beliebigkeit um Schätzungen.				500 kB	2 MB	4 GB

2.1.1.S Speicher des MathClient

Tabelle 2.3

Speicher	Struktur	Datentyp	Größe	Minimum	Mittel	Maximum
XML-Baum	Implementationsabhängig	Knotenobjekt mit Unterknoten (CObjectList)	Anzahl Knoten * 70 + (Anzahl Knoten – 1) * 4	70	50 kB	4 GB
k * Reele_Zahl	<zahl>	CReell	$\left\lceil \frac{\log_2 10^{\text{Stellenzahl}} + 1}{32} \right\rceil * 4 + 50$	k * 51	k * 500 Byte	k * 850 MB
Der **Gesamtbedarf** kann auf Grund der gewählten Betriebssysteme niemals 4 GB überschreiten. Bei den Werten handelt es sich wegen Beliebigkeit um Schätzungen.				500 kB	3 MB	4 GB

2.1.2 Weitere Datenflüsse

Es treten keine weiteren Datenflüsse auf.

2.1.3 Abschätzungen

- **Gesamtsystem:** Es wird von einer erwarteten Nutzungsdauer von 3 Jahren und einer maximalen Nutzungsdauer von 10 Jahren ausgegangen.

- **Benutzer**: Es wird von mindestens einem und maximal 1000 gleichzeitigen Anfragen (durch Benutzer und auch durch automatisierte Applikationen) am Server ausgegangen. Je schneller die Netzwerkverbindung und der Prozessor des Servers sind und je mehr Arbeitsspeicher zur Verfügung steht, desto mehr Benutzer können gleichzeitig bei akzeptablen Verzögerungs- und Rechenzeiten vom Server bedient werden.
  Am Klienten arbeitet stets genau ein Benutzer.

- **Serveranfragen:** Der Server wird 24 Stunden am Tag betrieben und steht deshalb immer zur Verfügung. Wenn ein Auftrag die Bearbeitungsdauer von einer Minute nicht überschreitet bei durchschnittlich 10 Aufträgen gleichzeitig können an einem Tag maximal 14400 Rechenanfragen befriedigt werden. Im Jahr wären dies bis zu 5 256 000 Anfragen.

- **Serverapplikationen:** Gegenwärtig ist vorgesehen, die Serverapplikation auf 3 leistungsstarken Großrechnern mit Windows XP Betriebssystem zu installieren.

- **Klientenapplikationen:** 200 Klienteninstallationen sind derzeit vorgesehen.

- **Netzwerk:** Es wird erwartet, dass der Serverrechner bei Leerlauf mit weniger als 70 ms Verzögerung von jedem Klientenrechner aus erreicht werden kann. Dann gelten die im Punkt Benutzer getroffenen Abschätzungen.

- **Eingabetext:** Ein Standardbenutzer wird einen Eingabetext von nicht mehr als 1024 Zeichen eingeben. Wird dieser in XML zur Datenübertragung umgewandelt, so sollte er nicht mehr als 5120 Zeichen (10 kB) enthalten. Diese werden im lokalen Netzwerk mit mindestens 10 MBit/s bzw. im Internet mit mindestens 56 kBit/s übertragen. Die Antwort des Servers sollte ebenfalls nicht mehr als den dreifachen Umfang der Anfrage haben (auf Grund der hohen Präzision der numerischen Daten). Dadurch entsteht ein geschätzter Gesamtumfang der Datenübertragung von 40 kB. Im Netzwerk würde die gesamte Datenübertragung weniger als 1 s und im Internet etwa 6 Sekunden dauern. Nach dem Punkt Serveranfragen übersteigt die reine Rechenzeit die Übertragungszeit also um das Zehn- bis Sechzigfache. Dazu kommen Zeiten für Kodieren und Dekodieren in und von XML.

- **Ausgabetext:** Aus dem Punkt Eingabetext folgt bereits, dass der Ausgabetext selten mehr als 3072 Zeichen umfassen wird.

- **XML-Baum:** Der XML-Baum wird, wie gesagt, durch Objekte dargestellt. Er wird im Normalfall nicht mehr als 100 Knoten umfassen und somit, in dekodierter Form ca. 50 kB einnehmen. Da man ihn nur rekursiv aus den Eingabedaten bestimmen kann, wird hier auch ein Anteil Rechenzeit benötigt. Wir schätzen diese auf 3 ms. Die rückwärtige Erstellung des Baums für den Aufbau Ausgabedaten jedoch erfolgt wesentlich schneller, da hier die verschiedenen Konversions-, Test- und Verifikationsalgorithmen entfallen. Dies führt zu einer Abschätzung der Rechenzeit auf 0,5 ms. Der Speicherbedarf ist hierbei gleich wenn nicht sogar größer, wegen der große mathematischen Genauigkeit entstehen wieder lange Zeichenketten. Zu beachten ist, dass beide Abläufe sowohl im Klienten als auch im Server realisiert werden müssen.

- **Reelle_Zahl:** Reelle Zahlen werden wie gesagt ebenfalls als Objekte mit Binärdaten in großem Umfang dargestellt. Hier ist der größte Speicherbedarf im Server zu suchen (nur dort existieren sie) und wird auch die meiste Rechenzeit aufgewendet. 59 Sekunden einer Rechenaufgabe werden mit der Kodierung, Dekodierung und Rechnung mit den Speichern vom Typ Reelle_Zahl verbracht. Es liegen noch keine algorithmischen Effizienzbetrachtungen hierzu vor, jedoch können wir bereits abschätzen, dass Operationen wie +, -, *, / mit sehr hoher Performance realisierbar sind, bei solchen jedoch, die Reihenentwicklungen benötigen (wie Arcsin, e^x oder auch die Konstante π) ist mit Abstrichen in der Operationsgeschwindigkeit zu rechnen.

2.1.4 Genauigkeit

Die Genauigkeit ist der springende Punkt des Projekts. Es soll ermöglicht werden, mathematische Operationen mit nahezu beliebiger Genauigkeit durchzuführen (siehe Punkt 2.1). Natürlich ist es weiterhin möglich, dass eine Operation den eingestellten Wertebereich überläuft, jedoch kann der Benutzer die Aufgabe dann einfach mit einem erweiterten Wertebereich noch einmal rechnen. Auch Rundungs- und Genauigkeitsfehler können weiterhin auftreten, jedoch werden sowohl mathematische und statistisch-stochastische Betrachtungen als auch interne Wertebereichsverbreiterungen durchgeführt, um dies zu umgehen. Das Projekt konzentriert sich in seiner Entwicklungsarbeit zu einem großen Teil auf die mathematische Genauigkeit.

2.2 Operationelle Anforderungen an die Datenströme

Im System treten physische Datenströme in teilweise mittlerem bis großen Umfang zwischen der Klienten- und der Serverapplikation auf. Sie können Spitzen von mehreren Megabytes erreichen, werden sich im Mittel gemäß 2.1.3 Abschätzungen im Kilobyte-Bereich bewegen. Im Vollbetrieb der Applikation können die Datenströme bei Mehrklientenbenutzung auch kritischen und überkritischen Umfang annehmen, den der Server physisch nicht mehr bewältigen kann. Aus diesem Grund wird ein sehr leistungsstarker Rechner mit ebenso leistungsstarker Netzwerkanbindung empfohlen. Bei Eintritt in den überkritischen Bereich der Netzwerkauslastung wird das Verhalten bei der Anfrageannahme stochastisch bzw. instabil. Die Serverapplikation an sich bleibt jedoch in stabilem Zustand, angenommene Aufträge werden korrekt abgearbeitet und lediglich mit Verzögerung zugestellt. Brechen Verbindungen unvermittelt ab, so werden sie der TCP/IP-Spezifikation und deren Implementierung in WINSOCK gemäß als nicht definiert offen gehalten und erst bei Neustart des Servers bzw. der Applikation geschlossen.

2.3 Operationelle Anforderungen an die Prozesse

Antwortzeiten von einer Minute werden als Maximum für umfangreiche Berechnungen bzw. bei hoher Serverbelastung akzeptiert, bei Standardberechnungen und normaler Netzlast werden jedoch 5 bis 10 Sekunden angestrebt.
Wird das Netzwerk bzw. der Server überlastet, bzw. verfügt der Klient nur über eine zu leistungsschwache Netzanbindung kann es zu Abbrüchen oder unerwartet langen Verzögerungszeiten kommen.

3 Spezifikation wichtiger Qualitätsanforderungen

3.1 Benutzbarkeit

Die Nutzer des Systems gehören entweder der Nutzergruppe Administrator oder der Nutzergruppe Benutzer an. Für den Administrator werden umfassende Kenntnisse im Umgang mit Computern und deren Software sowie Kenntnisse im Bereich Netzwerke vorausgesetzt. Hingegen werden für den Benutzer keine solch umfassenden Kenntnisse vorausgesetzt. Diese Nutzergruppe sollte lediglich im Umgang mit dem Betriebssystem Windows vertraut sein und grundlegende mathematische Kenntnisse vorweisen. Die Bedienung wird über eine leicht überschaubare, leicht verständliche und leicht erklärbare grafische Oberfläche erfolgen.
Da die Software MathServer für den Einsatz an Hochschulen und Universitäten vorgesehen ist, sollten diese Voraussetzungen allerdings von nahezu allen Benutzern erfüllt werden können.

3.2 Zuverlässigkeit

Da die Nutzer sehr hohe Anforderungen an die Zuverlässigkeit des Systems stellen, muss besonderes Augenmerk auf diesen Gesichtspunkt gelegt werden.
Um Bedienfehler von vornherein zu minimieren, wird die Bedienung, wie schon bei 3.1 erwähnt, über eine leicht überschaubare, leicht verständliche und leicht erklärbare grafische Oberfläche erfolgen. Mathematisch falsche Eingaben durch den Nutzer können vom Klienten erkannt werden. Diese Daten werden nicht zum Server übertragen. Der Nutzer wird durch eine spezifische Fehlermeldung auf den Eingabefehler hingewiesen und kann diesen korrigieren.
Des Weiteren muss man Übertragungsfehler im Netzwerk und andere Fehlfunktionen der Hard- bzw. Software zwischen Klient und Server in Betracht ziehen. Der Server ist in der Lage viele davon zu erkennen, allerdings wird es nicht möglich sein, alle Fehler festzustellen. Wenn ein solcher Übertragungsfehler erkannt wurde, wird der Benutzer durch eine Fehlermeldung darauf hingewiesen. Selbständige Fehlerkorrektur des Systems ist nicht möglich.
Jegliche Speicherdefekte können vernachlässigt werden, da im System keine Datenspeicherung vorgesehen ist.

3.3 Integrität

Auf Grund der Funktion und den Einsatzbedingungen sind für das System Mathserver ist keine besonderen Schutzmechanismen vorgesehen. Das Programm empfängt, bearbeitet, speichert oder versendet keinerlei Daten, die einer Geheimhaltung unterliegen könnten. Es werden nur mathematische Aufgaben empfangen, bearbeitet und deren Ergebnis (evtl. mit Zwischenergebnissen) zurückgesendet. Deshalb ist ein besonderer Schutz der Daten hinfällig.
Des Weiteren soll das System in Hochschulen und Universitäten zum Einsatz kommen. Das heißt, der Server befindet sich innerhalb eines Netzwerkes, dass durch Firewalls geschützt ist. Dies sollte einen zusätzlichen Schutz vor unberechtigten Zugriffen von außen auf das System gewährleisten.

3.4 Flexibilität

Da von Seiten der Benutzer hohe Anforderungen an die Erweiterbarkeit, die Flexibilität und die Zukunftssicherheit des Systems gestellt werden, wurde auf diesen Punkt besonderes Augenmerk gelegt.
Das System Mathserver arbeitet mit dem neuen und zukünftigen Textstandard – Unicode. Es unterstützt allerdings auch ältere Standards wie ASCII und ANSI. Die Eingabe und die Bearbeitung der mathematischen Formeln ist daher prinzipiell mit jedem Schreibprogramm möglich, dass eine dieser Standards unterstützt.
Des Weiteren wird der neue und zukünftige Standard für Datenversand und –Speicherung – XML – verwendet. Für den Umgang mit den mathematischen Formeln wird der Standard für mathematische Formeln – MATH ML – verwendet. Hierbei handelt es sich um eine spezielle XML – Anwendung. Die Daten für das Programm können mit jeder beliebigen XML bzw. MATH ML Applikation erstellt und editiert werden. Sowohl bei XML als auch bei MATH ML handelt es sich um internationalisierte, plattformunabhängige Standards.
Der Versand bzw. der Empfang der Daten wird http kompatibel organisiert. Deshalb können die Daten mit jeder beliebigen HTTP-fähigen Applikation übertragen werden. Es ist auch möglich, Daten direkt an einen Webbrowser zu schicken, sofern dieser XML bzw. MATH ML unterstützt.
Für die Übertragung der Daten wird das Standardprotokoll zur Datenübertragung – TCP/IP – verwendet. Dieses Protokoll wird, wie die meisten genutzten Standards, von allen üblichen Plattformen unterstützt. Diese Tatsachen garantieren ein Höchstmaß an Flexibilität.

3.5 Portabilität

Die Portabilität des Servers ist ziemlich eingeschränkt, da dieser Teil des Programms für das Betriebssystem Windows entwickelt wird. Der Server verwendet direkt Funktionen der Windows API, da durch die Benutzung dieser Funktionen ein Höchstmaß an Leistungsfähigkeit erreicht werden kann. Und dieses Höchstmaß an Effizienz ist das Ziel des Projekts. Deshalb ist eine Konvertierung des Servers für andere Betriebssysteme nahezu unmöglich. Das System sollte auch auf neueren Windows Versionen als die mindestens geforderten Windows 2000 und Windows XP optimal lauffähig sein, da auf die Windows-Kompatiblität großen Wert gelegt wurde.
Da Großrechner für den Internetbetrieb oft mit Microsoft Windows Betriebssystemen genutzt werden, ist die Abbildung des Math-Server-Konzepts auf diese Umgebung nur logisch.
Hingegen wurde auf die Portabilität des Klienten großen Wert gelegt. Dieser Teil des Systems ist zwar für Windows konzipiert worden, ist aber ohne größeren Aufwand für andere Plattformen (z.B. Linux, Unix, Mac) konvertierbar, sofern diese Plattformen die genutzten Standards (HTTP, XML, MATH ML, TCP/IP) unterstützen.

4 Basismaschine und Entwicklungsumgebung

4.1 Basismaschine

4.4.1 Hardware

Die notwendige Grundausstattung für die am Math-Server beteiligten PCs muss laut Aufgabenspezifikation in 2 Klassen unterteilt werden. Für den Server wird ein wesentlich leistungsstärkerer PC vorausgesetzt als für den Client.

Für den Math-Client wird folgendes System vorausgesetzt:

- 400 MHz CPU
- 64 MB RAM
- 20 MB freier Festplattenspeicher
- CD-ROM Laufwerk
- Netzwerkzugang (LAN), min. 10 MBit/s bzw. Internetzugang, Modem, 56 KBit/s
- Monitor 15'', 70 Hz und Grafikkarte, 256 Farben bei einer Auflösung von 800x600
- Tastatur, Maus

Für den Math-Server wird folgendes System vorausgesetzt:

- 400 MHz CPU
- 64 MB RAM
- 30 MB freier Festplattenspeicher
- CD-ROM Laufwerk
- Netzwerkzugang (LAN), min. 10 MBit/s bzw. Internetzugang, Modem, 56 KBit/s

Für den Math-Server wird folgendes System empfohlen:

- 1 GHz CPU
- 256 MB RAM
- 100 MB freier Festplattenspeicher
- CD-ROM Laufwerk
- Netzwerkzugang (LAN), 100 MBit/s bzw. Internetzugang, DSL

Mit den Systemsanforderungen für Server und Client lassen sich alle Anforderungen an die zu entwickelnde Software komplett erfüllen.
Die Daten für die vorausgesetzten Systeme sind für den Betrieb des Systems (sowohl Client als auch Server) mindestens notwendig damit das System einwandfrei funktioniert. Bei Nichteinhalten der Systemanforderungen kann der fehlerfreie Betrieb des Systems nicht garantiert werden.
Die Voraussetzungen für den Client und den Server sind nahezu deckungsgleich, da die Systemanforderungen des Gesamtsystems großteils durch die Systemanforderungen des geforderten Betriebssystems bestimmt werden.
Günstigere Bedingungen für den Betrieb des Servers lassen sich durch Nutzung des empfohlenen Systems erreichen.
Für den Client gibt es keine Systemempfehlung, da der Client bereits mit den Mindestvoraussetzungen optimal betrieben werden kann. Mit einem schnelleren System könnten keine günstigeren Bedingungen für den Betrieb geschaffen werden.

4.1.2 Betriebssystem

Als Betriebssystem wird sowohl für den Client als auch für den Server Windows 2000 oder Windows XP vorausgesetzt.
Die Betriebssystemvoraussetzungen begründen sich durch die Notwendigkeit der Unicode Unterstützung für die zu entwickelnde Software. Theoretisch kann das System auch mit der Microsoft Layer for Unicode auf niedrigeren Windowsversionen betrieben werden, einwandfreie Funktion kann jedoch nicht garantiert werden. Ohne systeminterne Unicodeunterstützung ist die Applikation nicht lauffähig.

4.1.3 Sonstige Basissoftware

Als sonstige Basissoftware ist Visual Basic Runtime Libraries für den Client notwendig.

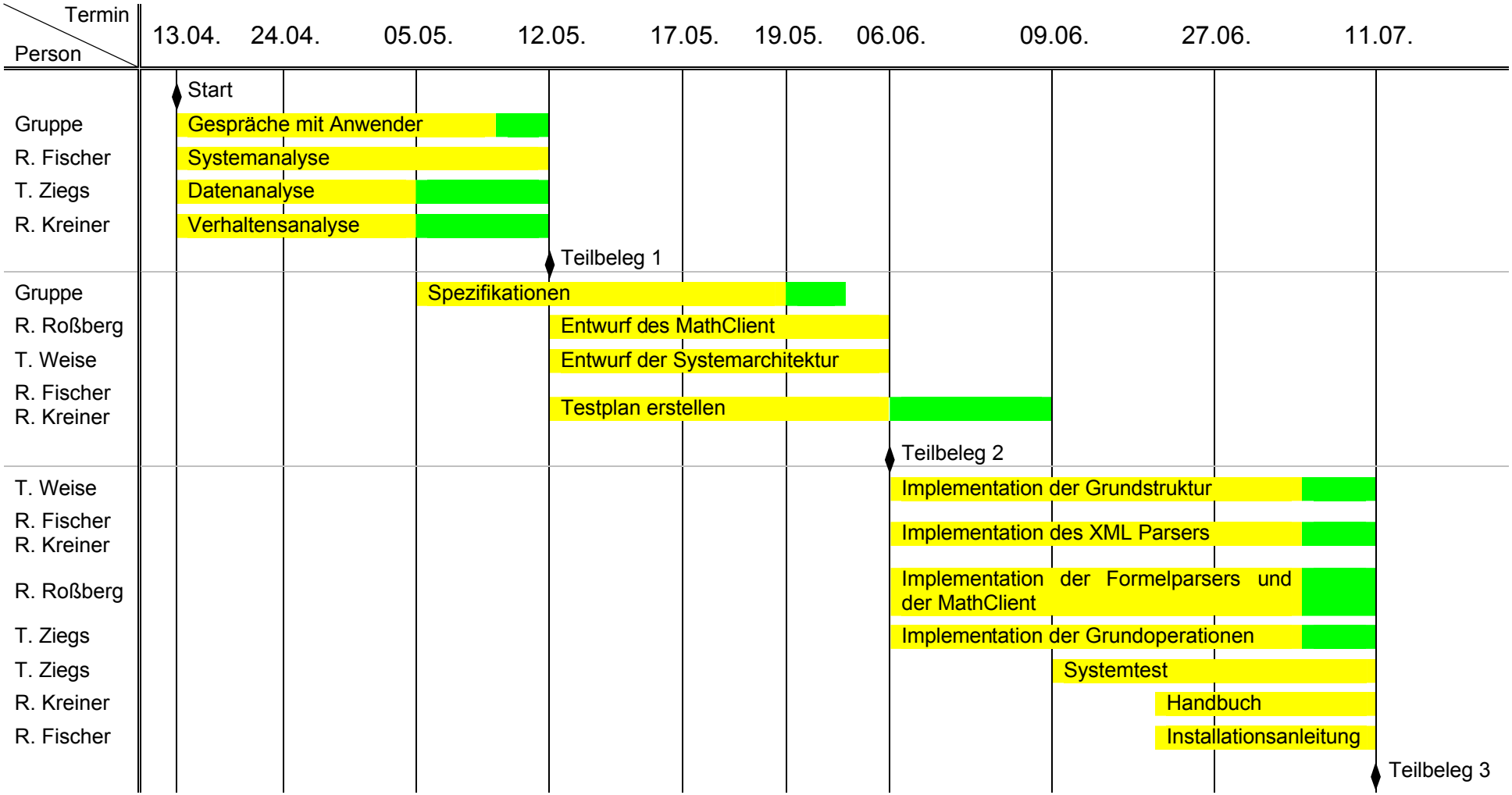
4.2 Entwicklungsumgebung

Als Entwicklungsumgebung ist das Microsoft Visual Studio 6.0 mit Servicepack 5.0 und das Microsoft Developer Network vorgesehen.
Als Rechner für Testung und Kompilierung werden Microsoft Windows 2000 bzw. XP auf Intel Pentium Maschinen 1 GHz aufwärts verwendet.

5 Planung

Die Entwicklung des Systems Math-Server erfolgt entsprechend des nachstehenden Plans.

Diagram 5.1 - Gantt-Diagramm



6 Teamplanung

Tabelle 6.1

Teammitglied	Aufgabenbereiche
Thomas Weise	Teamleiter, Hauptarchitekt, Entwurfsverantwortlicher Programmierer für Grundobjekte und Operationen und Zusammenfügen der einzelnen Module
Thomas Ziegs	Sachverständiger für Mathematik, Qualitätsicherungsmanagement Programmierer für sämtlichen mathematischen Operationen
Rico Roßberg	Dokumentationsspezialist, Eingabespezialist Programmierer des Klienten und des Formelparsers
Roland Fischer	Zeitplanverantwortlicher, Systemanalyst Programmierer des XML-Parsers und evtl. der Installationssoftware (in Unterteam)
René Kreiner	Verhaltensanalyst, Handbuch- und Hilfesystementwicklung Programmierer des XML-Parsers und evtl. der Installationssoftware (in Unterteam)

Softwarepraktikum 2003

MathServer

Projektdokumentation
(Teilbeleg 2)

Teamleiter: Thomas Weise
Mitglieder des Projektteams:
Roland Fischer
René Kreiner
Rico Roßberg
Thomas Ziegs

Praktikumsbetreuer : Dipl.-Inf. Lutz Neugebauer

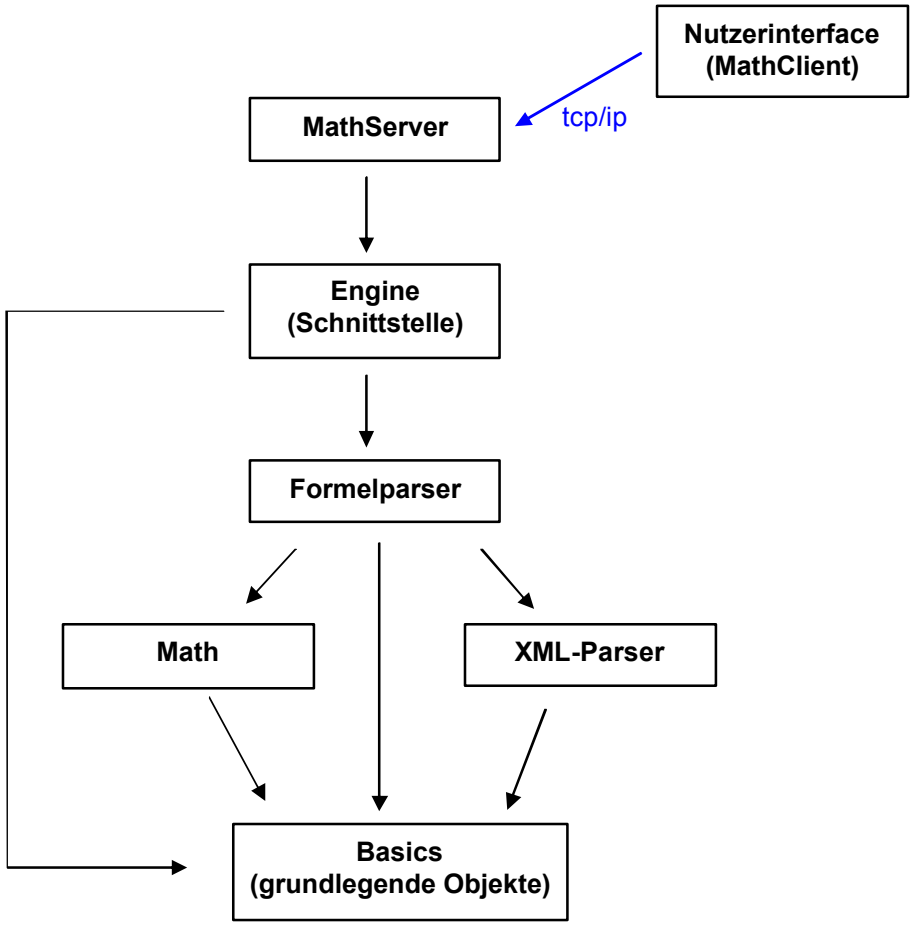
Chemnitz, den 23.06.2003

1 Entwurfsdokumentation

1.1 Entwurf der Systemarchitektur

1.1.1 Modulhierarchie der Bibliotheken

Diagramm 1.1



Wie man sehen kann, ist die Systemarchitektur sehr tief angelegt. Für die Module werden dynamische Linkbibliotheken (DLLs) genutzt, um bestmögliche Wartung zu gewährleisten.
Im Projekt wird der objektorientierte Programmieransatz weitflächig genutzt, die Klassen der Engine-Schnittstelle verwenden threadbasierte Techniken.
Die einzelnen Klassen der Applikation werden von den DLLs, welche stets von einer Arbeitsgruppe gemäß dem Plan entwickelt werden, exportiert.
Aufgrund der Vielzahl der Klassen und deren komplizierter Struktur ist es sinnvoller, die Module und Klassensysteme getrennt zu betrachten, wobei sich der globale Ausblick des Diagramms auf die DLLs konzentriert.

Die Module werden prinzipiell mit Visual C++ erstellt, nur das Nutzerinterface in Visual Basic. Es ist nicht möglich, normal exportierte Klassen vom C++ nach VB zu portieren, dies geht nur mit einzelnen Routinen oder ActiveX-Elementen. Da unser Projekt aber größtenteils auf simplen Klassen beruht, können sie nicht in den MathClient exportiert werden. Daher werden hier verschiedene Techniken separat implementiert.

1.2 Definition der Modulleistungen

Auf Grund des Umfangs und der Komplexität der Klassenbeziehungen innerhalb des Projekts wird von einer graphischen Darstellung abgesehen, stattdessen werden die Zusammenhänge geschildert.

Prinzipiell werden einige Definitionen zur Abkürzung von Speicherklassen verwendet. Dazu zählen VIR für virtuelle Methoden, SC für die Aufrufkonvention Standardcall und FC für die Aufrufkonvention Fastcall (= Pascal/Delphi-native). FRA steht für friend.

Parameter beginnend mit p_ sind Pointer auf den jeweiligen Datentyp, c_ Konstanten und r_ Parameter für das Call-By-Reference Aufrufschema.

Oftmals werden c_hresult Werte zurückgegeben, welche den Success- bzw. Fehlerkode der Operation repräsentieren.

1.2.1 Modul Basics

Modul:	Basics	**Bearbeiter:**	Thomas Weise
Bezeichner:	Basics	**Änderungsstand:**	28.05.2003

Kurzbeschreibung:

Das Modul Basics stellt als Grundlage aller anderen Module atomare Funktionen und Klassen zu Verfügung. Es importiert alle Windows-Routinen die im gesamten MathServer verwendet werden. Es werden keinerlei C-native Routinen angewandt, um maximale Performance zu gewährleisten basiert sämtliche Programmierarbeit auf der Windows API (Application Programming Interface) und selbst entwickeltem Code, vorzugsweise Assembler.

Alle Klassen aller anderer Module leiten sich von den im Modul Basics definierten Klassen ab.

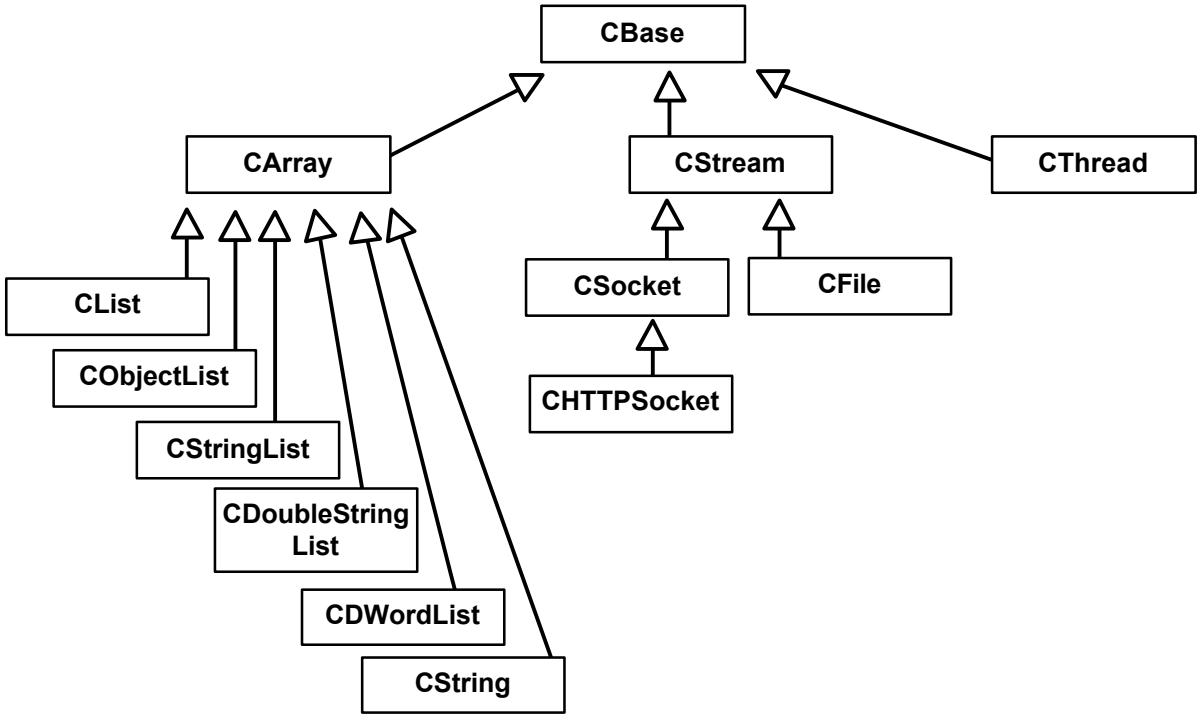
Export:
- CArray
- CBase
- CDoubleStringList
- CDWordList
- CFile
- CHTTPStream
- CList
- CObjectList
- CSocket
- CStream
- CString
- CThread
- gemeinsame Routinen für COM-Fehlercode Verarbeitung
- grundlegende mathematische Routinen (Fließkomma)
- Speicherzugriffsroutinen

Import: alle Windows API Routinen aus
- kernel32.dll
- user32.dll
- comdlg32.dll
- advapi32.dll
- shell32.dll
- ole32.dll
- oleaut32.dll
- mpr.dll
- ws2_32.dll

Auf dies wird hier nicht näher eingegangen.

1.2.1.1 Klassenvererbungsdiagramm

Diagramm 1.1



In diesem Diagramm stellen wir die Vererbung der Klassen des Moduls Basics dar.

1.2.1.2 Grundlegende Datentypen und Abkürzungen

Tabelle 1.1

Typ-Schema	Bedeutung
c_xxx	Konstante des Typs xxx
p_xxx	Zeiger auf Variable des Typs xxx
r_xxx	Call-By-Reference Parameter des Typs xxx
pc_xxx	Zeiger auf Variable des Typs c_xxx
cp_xxx	Konstantes des Typs p_xxx
pp_xxx	Zeiger auf Variable des Typs p_xxx

Datentype	Bedeutung
dword	32-Bit Zahl, vorzeichenlos
integer	32-Bit Zahl, mit vorzeichen
pointer	Zeiger
character	Unicode-Zeichen
asciichar	Ascii-Zeichen
string	Unicode-String-Daten
asciistring	Ascii-String-Daten
double	Fließkommazahl

1.2.1.3 CBase *(Klasse)* **Grundklasse**

Die Grundklasse, von der sich alle anderen Klassen ableiten, ist CBase.
Hier werden Grundfunktionen wie Referenzzählung implementiert.

Members	m_state	dword	32Bit für Zustand, Bit 31 reserviert: true wenn im Destruktor
	m_refcount	dword	Referenzzählung
	m_mutex	dword	evtl. Thread-Synchronisation

Methoden

Die folgenden drei Methoden sind für den Zugriff auf die Statusvariable des Objekts angelegt, hier können später beliebige boolesche Werte gespeichert werden. Bit 31 ist reserviert für die Verwendung ausschließlich in CBase (um Threadsicherheit der Referenzzählung zu gewährleisten). astate ist stets der zu setzende / testende Zustand, aset gibt an, ob er gesetzt oder gelöscht werden soll.

```
VIR c_dword    SC   State              (void);
VIR c_boolean  SC   GetState          (c_dword astate);
VIR c_dword    SC   SetState          (c_dword astate, c_boolean aset);
```

Die folgenden drei Methoden führen die Referenzzählung der Objekte durch, AddRef signalisiert eine neue Referenc, Release eine freigegebene Referenz.

```
VIR c_dword    SC   RefCount          (void);
VIR c_dword    SC   AddRef            (void);
VIR c_dword    SC   Release           (void);
```

Enter und Leave sind Methoden, die die Threadsicherheit gewährleisten. Sie finden in der Regel keinen direkten Gebrauch.

```
VIR void       SC   Enter             (void);
VIR void       SC   Leave             (void);
```

1.2.1.4 CArray *(Klasse)* und abgeleitete außer CString (abgeleitet von CBase)

Eine wichtige Gruppe der Klassen in unserem Projekt ist die derjenigen Klassen mit abzählbaren Unterelementen.

Das grundlegende Verhalten von diesen wird in CArray implementiert (leitet sich ab von CBase).
Die Nachkommen von CArray sind CDWordList (ein dynamischer Array von DWORDs (4 Byte)), CObjectList (ein dynamischer Array von Instanzen von CBase abgeleiteter Klassen), CStringList und CDoubleStringList (Listen von CStrings und CString-Paaren).
CList ist ein Array von Records beliebiger Größe, daher besitzt jedes Objekt noch einen Membereintrag für den Speicherbedarf der Einträge.

Es wird angenommen, dass die Einträge in den Arrays vom Typ ITEM sind.

Members	m_length	dword	Länge des Arrays (in Elementen)
	m_memlength	dword	Länge der reservierten Speicherstelle (in Elementen)
	m_data	p_ITEM	Zeiger auf reservierte Speicherstelle

Methoden

Die folgenden Methoden wurden direkt in CArray implementiert.

Setzt die Länge der Liste auf alength, wird in der Regel nicht direkt aufgerufen. Verwendet einen Wachstumsalgorithmus um die Geschwindigkeit zu optimieren.

```
VIR c_hresult   SC   SetLength        (c_dword alength);
```

Gibt die Länge des Arrays wieder.

```
VIR c_dword     SC   Length           (void);
```

Löscht den Inhalt der Liste.

```
VIR c_hresult   SC   Clear            (void);
```

Die folgenden Routinen wurden Klassenabhängig implementiert, d.h. dass z.B. eine CStringList die Methoden mit anderen Parametern/Rückgabewerten implementiert. Die jeweilige Konfiguration ist dem entsprechenden Header-File zu entnehmen. Das prinzipielle Verhalten ist jedoch stets gleich.

Gibt einen Zeiger auf den Speicher zurück, in dem die Arrayeinträge abgelegt sind.

```
VIR cp_ITEM     SC   DataPtr          (void);
```

Fügt einen Eintrag (aitem) der Liste hinzu, gibt optional dessen Index in aindex zurück.

```
VIR c_hresult   SC   Add              (c_ITEM aitem, p_dword aindex = 0);
```

Setzt den Eintrag mit dem Index aindex auf den Wert aitem, vergrößert die Liste wenn notwendig.

```
VIR c_hresult   SC   SetAt            (c_dword aindex, c_ITEM aitem);
```

Liefert den Eintrag mit dem Index aindex in aitem zurück.

```
VIR c_hresult   SC   GetAt            (c_dword aindex, r_ITEM aitem);
```

Löscht den Eintrag mit dem Index aindex.

```
VIR c_hresult   SC   Delete           (c_dword aindex);
```

Fügt an der Stelle aindex den Eintrag aitem ein.

```
VIR c_hresult   SC   Insert           (c_dword aindex, c_ITEM aitem);
```

Sucht nach einem Eintrag mit dem Wert von aitem, liefert dessen Index (bzw. NOT_FOUND) zurück.

```
VIR c_dword     SC   Find            (c_ITEM aitem);
```

Entfernt alle Einträge mit dem Wert aitem.

```
VIR c_hresult   SC   Remove          (c_ITEM aitem);
```

1.2.1.5 CString *(Klasse)* (abgeleitet von CBase)

CString repräsentiert eine Zeichenkette beliebige Länge. Diese wird im Speicher als Unicodestring aufbewahrt. CString leitet sich ebenfalls von CArray ab, verfügt jedoch über eine viel weitere Methodenpalette.

Die **Members** der Klasse entsprechen der Definition von CArray.

Methoden

Auch die Methoden SetLength, Length, DataPtr und Clear entsprechen denen von CArray. GetAt und SetAt arbeiten mit Unicodezeichen.

Viele Methoden können entweder mit ASCII- oder Unicode-Strings bzw. CString-Objekten als Parameter aufgerufen werden. Wir stellen dies durch ein TYP im Namen dar, was entsprechen durch Ascii, Unicode oder Str ersetzt werden müsste. Bei Ascii-Funktionen gibt es stets den zusätzlichen Parameter acodepage, der den Index einer zur Übersetzung zu verwendenden Codepage enthält.
Ähnliches Verhalten gilt für Zeichenparameter, hier ersetzt CHR entweder Unichar oder Asciichar.
Für die jeweiligen Parameter schreiben wir DAT als Datentyp.
Dadurch wird es möglich, die Funktionen mit beliebiger Zeichenbreite einzusetzen. Speziell für die Dekodierung der empfangenen/zu senden Daten wurden From- und ToBuffer-Routinen entwickelt, die sowohl ASCII- als auch Unicode mit sowohl Big- und LittleEndian einlesen und schreiben kann. Nach XML-Spezifikation ist dies nämlich zulässig.

Für einige Funktionen (die Suchoperationen implementieren) stet jeweils eine Version zur Verfügung, die vom Beginn des Strings sucht und eine, die am Ende beginnt. Dies kürzen wir mit FFF ab, welches dann entweder durch nichts oder FE (From End) ersetzt wird.

Berechnet die Länge des Strings neu und passt sie an, wenn aresizeifpossible wahr ist.

```
VIR c_hresult  SC RecalculateLength  (c_boolean aresizeifpossible =
                                      true);
```

Kopiert den Quellstring astring, und zwar maximal amaxlen Zeichen in den String.

```
VIR c_hresult  SC FromTYP            (DAT     astring,
                                      c_dword amaxlen = MAX_DWORD);
```

Fügt maximal amaxlen Zeichen von astring an den String an.

```
VIR c_hresult  SC AddTYP             (DAT     astring,
                                      c_dword amaxlen = MAX_DWORD);
```

Fügt ein „Line-Feed“ (XML-Definition des Zeilenumbruchs) an den String an.

```
VIR c_hresult  SC Return             (void);
```

Fügt ein Tabulatorzeichen an den String an.

```
VIR c_hresult  SC Tab                (void);
```

Fügt ein Leerzeichen an den String an.

```
VIR c_hresult  SC Space              (void);
```

Fügt das Zeichen achar an den String an.

```
VIR c_hresult  SC AddCHR             (c_DAT achar);
```

Fügt maximal amaxlen Zeichen von astring an Stelle aposition ein.

```
VIR c_hresult  SC InsertTYP          (c_DAT   astring,
                                      c_dword aposition = 0,
                                      c_dword amaxlen = MAX_DWORD);
```

Löscht alength Zeichen aus dem String, beginnend bei aindex.

```
VIR c_hresult  SC Delete             (c_dword aindex = 0,
                                      c_dword alength = 1);
```

Vergleicht den String mit einem anderen (astring); man kann festlegen, ob Groß- und Kleinschreibung unterschieden (acasesensitive) wird, und wieviele (amaxlen) Zeichen verglichen werden sollen.

```
VIR c_dword    SC CompareTYP         (c_DAT     astring,
                                      c_boolean acasesensitive = false,
                                      c_dword   amaxlen = MAX_DWORD);
```

Sucht astring, beginnend bei astart, indem stets maximal amaxlen Zeichen von astring verglichen werden. acasesensitive gibt ab, ob Groß- und Kleinschreibung unterschieden wird.

```
VIR c_dword    SC FindTYPFFF          (c_DAT     astring,
                                      c_dword   astart = 0,
                                      c_boolean acasesensitive = false,
                                      c_dword   amaxlen = MAX_DWORD);
```

Löscht maximal atimes Auftreten von astring, beginnend ab astart unter acasesensitive Beachtung der Goß- und Kleinschreibung durch Vergleich von je maximal amaxlen Zeichen.

```
VIR c_hresult   SC RemoveTYPFFF       (c_DAT     astring,
                                       c_dword   atimes = MAX_DWORD,
                                       c_dword   astart = 0,
                                       c_boolean acasesensitive = false,
                                       c_dword   amaxlen = MAX_DWORD);
```

Löscht alle Zeichen vor anewstart und nach anewend.

```
VIR c_hresult   SC DeleteFrame        (c_dword anewstart,
                                       c_dword anewend);
```

Löscht alle führenden und folgenden Leer- und Steuerzeichen.

```
VIR c_hresult   SC Trim               (void);
```

Löscht alle Steuerzeichen, auch doppelte Leerzeichen bei aallowdoublespaces falsch.

```
VIR c_hresult   SC Clean              (void);
```

Kopiert alle Zeichen zwischen astart und aend in einen neuen String astring.

```
VIR c_hresult   SC Extract            (c_dword astart,
                                       c_dword aend,
                                       CString*& astring);
```

Vervielfacht den String um atimes durch Anhängung.

```
VIR c_hresult   SC Multiply           (c_dword atimes);
```

Erstellt eine ASCII-Repräsentationastring des Strings, indem alle Zeichen zwischen astart und aend mit Hilfe von acodepage transformiert werden.

```
VIR c_hresult   SC AsAscii            (asciistring& astring,
                                      c_dword acodepage = CP_THREAD_ACP,
                                       c_dword astart = 0,
                                       c_dword aend = MAX_DWORD);
```

Transformiert den String zu Großbuchstaben.

```
VIR void        SC Upper              (void);
```

Transformiert den String zu Kleinbuchstaben.

```
VIR void        SC Lower              (void);
```

Erstellt den String aus den Daten eines Global Unique Idendifiers (aguid).

```
VIR c_hresult   SC FromGuid             (c_guid aguid);
```

Schreibt die Stringdaten in einen GUID (aguid), wenn der String bereits wie ein GUID formatiert ist.

```
VIR c_hresult   SC AsGuid               (r_guid aguid);
```

Löst einen lokalen Pfad über das Netzwerk zu einem Universellen Pfad auf.

```
VIR c_hresult   SC MakePathUniversal    (void);
```

Erstellen einen Systemeindeutigen Namen für eine temporäre Datei, und löscht diese wenn adeleteprobfile wahr ist.

```
VIR c_hresult   SC GetTempFileName   (c_boolean adeleteprobefile = false);
```

amaxlendel Zeichen von adelete werden maximal atimes durch maximal amaxlenins Zeichen von ainsert unter acasesensitive Beachtung der Groß- und Kleinschreibung ersetzt.

```
VIR c_hresult   SC ReplaceTYPFFF     (c_DAT     adelete,
                                      c_DAT     ainsert,
                                      c_dword   atimes = MAX_DWORD,
                                      c_dword   astart = 0,
                                      c_boolean acasesensitive = false,
                                      c_dword   amaxlendel = MAX_DWORD,
                                      c_dword   amaxlenins = MAX_DWORD);
```

Ein sprint-Pendant.

```
VIR c_hresult   SC FormatTYP         (c_DAT    aformat,
                                      c_pointer aparams);
```

Fügt einen sprintf-ähnlich formatierten String an.

```
VIR c_hresult   SC AddFormatTYP      (c_DAT    aformat,
                                      c_pointer aparams);
```

Überprüft ob ein String mit astart beginnt, indem maximal amaxlen Zeichen unter Berücksichtigung von acasesensitive (s.o.) überprüft werden.

```
VIR c_boolean   SC StartsWithTYP     (c_DAT    astart,
                                      c_boolean acasesensitive = false,
                                      c_dword amaxlen = MAX_DWORD);
```

Überprüft ob ein String mit aend endet, indem maximal amaxlen Zeichen unter Berücksichtigung von acasesensitive (s.o.) überprüft werden.

```
VIR c_boolean   SC EndsWithTYP       (c_DAT    aend,
                                      c_boolean acasesensitive = false,
                                      c_dword amaxlen = MAX_DWORD);
```

Wendet die Windows Stringtypinformationssuche auf die Zeichen zwischen astart und aend an, gibt das Ergebnis in aresult zurück.

```
VIR c_hresult   SC StringType        (r_stringtype aresult,
                                      c_dword      astart = 0,
                                      c_dword      aend = MAX_DWORD);
```

Erstellt/schreibt den String aus/in einen Puffer, wobei die Kodierung ermittelt/gesetzt wird.

```
VIR c_hresult   SC FromBuffer        (c_pointer  abuffer,
                                      c_dword    asize,
                                      r_dword    aencoding);
VIR c_hresult   SC ToBuffer          (r_pointer  abuffer,
                                      r_dword    asize,
                                      c_dword    aencoding,
                                      c_boolean aputtrailingzero = false);
```

Formatiert einen String mit der Windows FormatMessage-Routine.

```
VIR c_hresult   SC FormatMessageTYP     (c_dword   aflags,
                                         c_DAT     asource,
                                         c_dword   amessageid,
                                         c_pointer aarguements = 0,
                                   c_dword alanguageid = DEFAULT_LANGUAGE);
```

Wandelt einen String von/zu einer Windows/Berkeley Netzwerkadressenstruktur (ainternetaddress) um, wobei Namensauflösung betrieben wird. FormLocalIP verwendet die lokale IP-Adresse.

```
VIR c_hresult   SC ToIPAddress       (r_sockaddr_in ainternetaddress);
VIR c_hresult   SC FromIPAddress     (c_sockaddr_in ainternetaddress);
VIR c_hresult   SC FromLocalIP       (c_word aport = 0);
```

Wandelt eine 8-Byte Zahl in einen String um, der hinzugefügt wird, bzw. erstellt eine neue Nummer aus dem String (bzw. einem Teil des Strings von astart bis aend).

```
VIR c_hresult   SC AddNum            (c_qword x);
VIR c_hresult   SC FromNum           (c_qword x);
VIR c_hresult   SC ExtractNum        (r_qword x,
                                      c_dword astart = 0,
                                      c_dword aend = MAX_DWORD);
```

Spiegelt den String (bzw. alle Zeichen zwischen astart und aend).

```
VIR void        SC Rotate            (c_dword astart = 0,
                                      c_dword aend = MAX_DWORD);
```

Sucht das Ende eines gequoteten Textes (entweder mit “ oder ’), wobei beliebige Schatelungen der Quotes beachtet werden, beginnen bei astart.

```
VIR c_dword     SC QuoteEnd          (c_dword astart);
```

Untersucht, ob astring an der Stelle apos steht (maximal amaxle Zeichen bzw. Groß-/Kleinschreibung gemäß acasesensitive).

```
VIR c_boolean   SC IsAtTYP           (c_DAT     astring,
                                      c_dword   apos,
                                      c_boolean acasesensitive = false,
                                      c_dword   amaxlen = MAX_DWORD);
```

1.2.1.6 CStream *(Klasse)* und abgeleitete (abgeleitet von CBase)

Neben CArray (Deszendent von CBase) und ihren Nachfahren spielen die Streams (CStream) eine große Rolle. Um möglichst variablen Umgang mit Eingangs- und Ausgangsdaten realisieren zu können, wird ein Grundobjekt mit den minimalen Methoden um Daten zu lesen und zu schreiben zur Verfügung gestellt. Es wird bereits hier das Verhalten zum Lesen und Schreiben von CString-Objekten festgelegt. Abgeleitet von CStream wird einerseits CFile als Repräsentant einer normalen Datei auf der Festplatte, andererseits CSocket, was ein Windows (Berkley Befehlssatz) Socket kapselt. Spätere Routinen zum Daten empfangen und versenden werden lediglich CStream verwenden, so das darunter liegende Logik das verwendete Medium gar nicht berücksichtigen muss. Von CSocket wird auch CHTTPSocket abgeleitet, so dass ebenfalls die Verwendung des HTTP-Protokolls ermöglicht wird.

Members Members sind entsprechende Windows bzw. Socket Handles bzw. Adress- und Steuerinformationsdaten ohne näheres Interesse für den Nutzer.

Methoden

Alle der genannten Klassen implementieren (überschreiben) die folgenden Routinen, teilweise werden neue als Supersets hinzugefügt, dem jeweiligen Anwendungsgebiet entsprechend.
Dazu gehört eine Open-Routine für CFiles bzw. Client/Server-Festlegungen für CSockets. Diese sind jedoch beim generischen Verhalten der Streams prinzipiell zu ignorieren.

Überprüft, ob der Stream für Datentransaktionen geöffnet ist.

```
VIR c_boolean  SC IsOpen            (void);
```

Überprüft, ob der Stream für Leseoperationen geöffnet ist.

```
VIR c_boolean  SC CanRead           (void);
```

Überprüft, ob der Stream für Schreiboperationen geöffnet ist.

```
VIR c_boolean  SC CanWrite          (void);
```

Schließt den Stream für alle Operationen.

```
VIR c_hresult  SC Close             (void);
```

Schreibt asize Bytes von adata in den Stream und gibt in awritten die tatsächlich geschriebene Anzahl zurück.

```
VIR c_hresult  SC Write             (c_pointer adata,
                                     c_dword   asize,
                                     r_dword   awritten);
```

Liest asize Bytes aus dem Stream in adata und speichert die tatsächlich gelesene Anzahl Bytes in aread.

```
VIR c_hresult  SC Read              (c_pointer adata,
                                     c_dword   asize,
                                     r_dword   aread);
```

Schreibt den String astring unter Berücksichtigung von aencoding in den Stream.

```
VIR c_hresult  SC WriteStr          (p_CString astring,
                                     c_dword   aencoding);
```

Liest astring aus dem Stream und speichert das angewandte Enkodierungsverfahren in aencodind.

```
VIR c_hresult  SC ReadStr           (p_CString astring,
                                     r_dword   aencoding);
```

1.2.1.7 CThread *(Klasse)* (abgeleitet von CBase)

Die letzte vom Basics Modul zur Verfügung gestellte Klasse ist CThread, was einen Windows Thread kapselt. Hier wurde jedoch ermöglicht, virtuelle Routinen als Thread-Routine zu nutzen. Zusätzlich wurde eine erweiterte Synchronisation implementiert, die es ermöglicht, vom Thread beliebige Unterthreads zu starten. Wird auf den übergeordneten Thread gewartet, so wird automatisch auch die Terminierung aller Unterthreads abgewartet. Dies ermöglicht es theoretisch, den Server in einer Instanz auf beliebig vielen Ports mit je beliebig vielen Verbindungen gleichzeitig ablaufen zu lassen.

Members	m_handle	dword	Handle zum Thread
	m_id	dword	ID des Threads
	m_owner	p_CThread	Übergeordneter Thread
	m_subthreadsevent	dword	Event für Kompletttermination der Subthreads
	m_exitcode	hresult	COM Success-/Fehlerkode

Methoden

Die Thread-Routine. Sie wird in allen abgeleiteten Klassen überschrieben und führt die “Aufgabe” des Threads durch.

```
VIR c_hresult SC Execute              (void);
```

Startet den Thread (zum ersten Mal oder nachdem er mit Suspend unterbrochen wurden).

```
VIR c_hresult SC Resume               (void);
```

Unterbricht die Ausführung des Threads.

```
VIR c_hresult SC Suspend              (void);
```

Teilt dem Thread mit, dass er die Ausführung von Schleifen unterbrechen und keine neuen Subthreads mehr starten soll, um terminieren zu können. (Automatisch auch Terminate aller Subthreads)

```
VIR c_hresult SC Terminate            (void);
```

Schickt das Terminate-Signal an alle Subthreads.

```
VIR c_hresult SC TerminateSubThreads  (void);
```

Wartet auf die Beendigung aller Subthreads.

```
VIR c_hresult SC WaitForSubThreads    (c_dword atime = INFINITE);
```

Wartet auf die Beendigung des Mainthreads.

```
VIR c_hresult SC WaitForMainThread    (c_dword atime = INFINITE);
```

Wartet, bis sowohl alle Subthreads als auch der Mainthread ihre Ausführung beendet haben.

```
VIR c_hresult SC WaitFor              (c_dword atime = INFINITE);
```

Sendet das Terminate-Signal an den Thread und wartet auf Beendigung.

```
VIR c_hresult SC TerminateAndWait     (void);
```

Gibt den Rückgabewert der Threadroutine Execute wieder, steht erst nach Terminate und Beendigung des Threads zur Verfügung.

```
VIR c_hresult SC ExitCode             (void);
```

Gibt die eindeutige ID des Threads wieder.

```
VIR c_dword   SC Id                   (void);
```

Gibt die Priorität des Threads wieder bzw. setzt sie.

```
VIR c_hresult SC SetPriority          (c_dword apriority);
VIR c_dword   SC GetPriority          (void);
```

Gibt wieder, ob der Thread gestartet wurde, gerade läuft, unterbrochen ist oder bereits beendet wurde.

```
VIR c_boolean SC Started              (void);
VIR c_boolean SC Running              (void);
VIR c_boolean SC Suspended            (void);
VIR c_boolean SC Terminated           (void);
```

Übergibt die Kontrolle an einen anderen Thread.

```
VIR void      SC SwitchToOther        (void);
```

1.2.2 Modul Engine

Modul: Engine **Bearbeiter:** Thomas Weise

Bezeichner: Engine **Änderungsstand:** 28.05.2003

Kurzbeschreibung:

Das Modul ist das Top-Level Modul des MathServer. Es importiert sämtliche Routinen von allen untergeordneten Modulen (siehe Diagramm 1.1).

Export: CServer **Import:** alle untergeordnete Module

1.2.2.1 CServer *(Klasse)* (abgeleitet von CThread)

Die Klasse CServer leitet sich direkt von CThread ab und verwendet ein CSocket. Jede Instanz von CServer arbeitet an einem Port (siehe Kommentar zu 1.2.1.5 CThread), und ermöglicht so einen sehr variablen Einsatz des MathServer. Das Hauptprogramm importiert lediglich die zugehörige DLL und erzeugt eine Instanz von CServer. Dann startet es einen MessageLoop (von 1.2.1 Basics importiert) und terminiert die Serverinstanz nach Ende des MessageLoops.

Members m_socket p_CSocket Instanz eines Server-Sockets

Methoden

Die überschriebene Execute-Routine von CThread akzeptiert ständig neue Klienten-Sockets.

VIR c_hresult SC Execute (void);

1.2.3 Modul Formelparser

Modul:	Formelparser	**Bearbeiter:**	Rico Roßberg
Bezeichner:	Formelparser	**Änderungsstand:**	05.06.2003

Kurzbeschreibung:

Wandelt die Eingabedaten vom MathServer (XML-String) mit Hilfe der untergeordneten Module in die Ausgabedaten (neuer XML-String) um.

Export:	Formula_Parse	**Import:**	Alle Leistungen der Module Basics, Math und XML-Parser

1.2.3.1 Formula_Parse (Funktion)

Der Formelparser erhält eine Zeichenfolge astring, die den im Nutzerinterface eingegebenen mathematischen Ausdruck enthält. Mit Hilfe des Moduls XML-Parser wird diese Zeichenfolge in einen XML-Baum umgewandelt, und mit Hilfe des Math-Moduls werden die End- und Zwischenergebnisse des mathematischen Ausdrucks berechnet. Diese werden zu einem neuen XML-Baum zusammengefügt, der vom XML-Parser zur Zeichenfolge aresult umgewandelt wird.

```
API c_hresult SC Formula_Parse (p_CString astring, p_CString aresult);
```

1.2.4 Modul Math

Modul:	Math	**Bearbeiter:**	Thomas Ziegs
Bezeichner:	Math	**Änderungsstand:**	28.05.2003

Kurzbeschreibung:

Das Modul Math stellt sämtliche mathematischen Routinen des Projekts zur Verfügung. Es ermöglicht die Berechnung mit beliebiger Stellenanzahl.

Export:	CConstants CReal	**Import:**	CBase Basics Routinen

1.2.4.1 CConstants *(Klasse)* (abgeleitet von CBase)

Die Objekte der Klasse CConstants halten ein Set von Konstanten für je eine Rechenanfrage. Da Zahlen wie π für Arcus Sinus benötigt werden, werden sie nur einmal berechnet, und dann gespeichert. Ihre Berechnung erfolgt, wenn sie das erste Mal benötigt werden.

Members	m_length	dword	Anzahl von DWord-Einheiten der Zahl
	m_dotpos	dword	logische Position des Kommas
	m_pi	pointer	Konstante π
	m_hpi	pointer	Konstante $\frac{\pi}{2}$
	m_fpi	pointer	Konstante $\frac{\pi}{4}$
	m_one	pointer	Konstante 1
	m_two	pointer	Konstante 2

Methoden

Gibt π wieder.

```
VIR c_pointer          PI                  (void);
```

Gibt $\frac{\pi}{2}$ wieder.

```
VIR c_pointer          Half_PI             (void);
```

Gibt $\frac{\pi}{4}$ wieder.

```
VIR c_pointer          FOURTH_PI           (void);
```

Gibt 1 wieder.

```
VIR c_pointer          ONE                 (void)
```

Gibt 2 wieder.

```
VIR c_pointer          TWO                 (void);
```

Gibt 4 wieder.

```
VIR c_pointer          TWO                 (void);
```

1.2.4.2 CReal *(Klasse)* (abgeleitet von CBase)

CReal kapselt die schnellen binärbasierten Rechenoperationen in eine Klasse, die vom FormelParser effektiv genutzt werden kann.

Members			
	m_length	dword	Anzahl von DWord-Einheiten der Zahl
	m_dotpos	dword	logische Position des Kommas
	m_userintbits	dword	Anzahl der Bits im Ganzzahlanteil
	m_userfracbits	dword	Anzahl der Bits im Nachkommaanteil
	m_error	boolean	Gibt an, ob die Zahl gültig ist
	m_data	p_dword	Daten der Zahl

Methoden

Setzt eine Zahl auf ungültig.

```
VIR void          SC   Invalidate           (void);
```

Gibt an, ob die Zahl ungültig ist

```
VIR boolean       SC   Error                (void);
```

Löscht die reelle Zahl (setzt sie auf undefiniert).

```
VIR c_hresult  SC Clear          (void);
```

Kopiert die reelle Zahl.

```
FRA cp_CReal   SC Copy           (cp_CReal  a1);
```

Bildet das Einerkomplement der reellen Zahl.

```
FRA cp_CReal   SC Not            (cp_CReal  a1);
```

Bildet das Zweierkomplement der reellen Zahl.

```
FRA cp_CReal   SC Not2           (cp_CReal  a1);
```

Gibt wieder, ob die Zahl negativ ist.

```
FRA c_boolean  SC Negative       (cp_CReal  a1);
```

Gibt wieder, ob die Zahl positiv ist.

```
FRA c_boolean  SC Positive       (cp_CReal  a1);
```

Gibt wieder, ob die Zahl 0 ist.

```
FRA c_boolean  SC IsZero         (cp_CReal  a1);
```

Addiert zwei reell Zahlen und speichert das Ergebnis in einer neuen.

```
FRA cp_CReal   SC Add            (cp_CReal  a1,
                                  cp_CReal  a2);
```

Subtrahiert zwei reell Zahlen und speichert das Ergebnis in einer neuen.

```
FRA cp_CReal   SC Sub            (cp_CReal  a1,
                                  cp_CReal  a2);
```

Multipliziert zwei reell Zahlen und speichert das Ergebnis in einer neuen.

```
FRA cp_CReal   SC Mul            (cp_CReal  a1,
                                  cp_CReal  a2);
```

Dividiert zwei reell Zahlen und speichert das Ergebnis in einer neuen.

```
FRA cp_CReal   SC Div           (cp_CReal  a1,
                                 cp_CReal  a2);
```

Erstellt eine neue Zahl aus einem String astring. Dabei besteht die Zahl aus alength Verarbeitungseinheiten, mit dem Komma an Stelle adotpos. Der Benutzer fordert auserintbits Vorkomma- und auserfracbits Nachkommastellen.

```
FRA   cp_CReal      SC   FromString        (c_dword       alength,
                                            c_dword       adotpos,
                                            c_dword       auserintbits,
                                            c_dword       auserfracbits,
                                            cp_CString    astring);
```

Schreibt die Zahl anum in den String astr.

```
FRA   c_hresult     SC   ToString          (cp_CReal      anum,
                                            cp_CString    astr);
```

Liefert den Vorkommateil der Zahl zurück.

```
FRA   cp_CReal      SC   Int               (cp_CReal      a1);
```

Liefert die Nachkommastelle der Zahl.

```
FRA   cp_CReal      SC   Frac              (cp_CReal      a1);
```

Eine Ganzzahldivision von a1 durch a2 durch und speichert das Ergebnis im Rückgabewert.

```
FRA   cp_CReal      SC   IDiv              (cp_CReal      a1,
                                            cp_CReal      a2);
```

Rest der Division von a1 durch a2.

```
FRA   cp_CReal      SC   Mod               (cp_CReal      a1,
                                            cp_CReal      a2);
```

Liefert den Betrag von a1.

```
FRA   cp_CReal      SC   Abs               (cp_CReal      a1);
```

Liefert die Fakultät von a1.

```
FRA   cp_CReal      SC   Factorial         (cp_CReal      a1);
```

Liefert e hoch a1.

```
FRA   cp_CReal      SC   E_X               (cp_CReal      a1);
```

Sinus von a1 in Bogenmaß.

```
FRA   cp_CReal      SC   Sin               (cp_CReal      a1);
```

Kosinus von a1 in Bogenmaß.

```
FRA   cp_CReal      SC   Cos               (cp_CReal      a1);
```

Tangenz von a1 in Bogenmaß.

```
FRA   cp_CReal      SC   Tan               (cp_CReal      a1);
```

Natürlicher Logarithmus von a1.

```
FRA   cp_CReal      SC   Ln                (cp_CReal      a1);
```

Arcussinus von a1 im Bogenmaß, wobei die Konstanten aus dem Set consts verwendet werden.

```
FRA   cp_CReal      SC   ArcSin            (cp_CReal      a1,
                                            cp_CConstants consts);
```

Arcuskosinus von a1 im Bogenmaß, wobei die Konstanten aus dem Set consts verwendet werden.

```
FRA   cp_CReal      SC   ArcCos            (cp_CReal       a1,
                                             cp_CConstants  consts);
```

Arcustangens von a1 im Bogenmaß, wobei die Konstanten aus dem Set consts verwendet werden.

```
FRA   cp_CReal      SC   ArcTan            (cp_CReal       a1,
                                             cp_CConstants  consts);
```

Die a2. Wurzel von a1.

```
FRA   cp_CReal      SC   Wurzel            (cp_CReal       a1,
                                             cp_CReal       a2);
```

a1 hoch a2.

```
FRA   cp_CReal      SC   Potenz            (cp_CReal       a1,
                                             cp_CReal       a2);
```

1.2.5 Modul XML-Parser

Modul:	XML-Parser	**Bearbeiter:**	Roland Fischer René Kreiner
Bezeichner:	XML_Parser	**Änderungsstand:**	28.05.2003

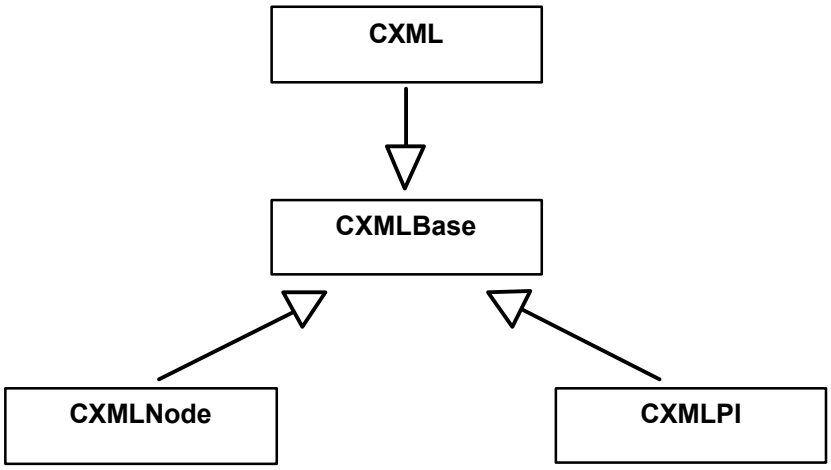
Kurzbeschreibung:

Das Modul XML-Parser dient der Umwandlung eines Strings (mit XML-Daten) in eine Baumstruktur zum gezielten Zugriff auf die einzelnen Knoten und ihrer Attribute. Ebenso wird das Aufbauen eines XML-Baumes unterstützt und das Generieren eines Strings aus diesem.

Export:	CXML CXMLBase CXMLNode CXMLPI xml_parse	**Import:**	CBase Basics Routinen

1.2.5.1 Klassenvererbungsdiagramm

Diagramm 1.2



1.2.5.2 CXML *(Klasse)* (abgeleitet von CXML)

CXML repräsentiert ein komplettes XML-Dokument. Es ist der Einstiegspunkt für das Modul Formelparser.
Die FromString, ToString, Clean und Type Funktionen werden entsprechen überschrieben.

Members m_items p_CObjectList Liste von XML-Objekten

Methoden

FromString erstellt ein XML-Objekt aus einem String (astring). Dabei werden die zu dem Objekt gehörenden Daten aus dem String gelöscht.

```
VIR c_hresult     SC   FromString          (cp_CString astring);
```

ToString veranlasst ein XML-Objekt, sich in einen String (astring) zu schreiben.

```
VIR c_hresult     SC   ToString            (cp_CString astring);
```

Type liefert den genauen Typ eines XML-Objekts als ganzzahligen Code zurück. (hier 1)

```
VIR c_dword       SC   Type                (void);
```

Get_Sub_Count liefert die Anzahl der Kindknoten eines XML-Objekts zurück.

```
VIR c_dword       SC   Get_Sub_Count       (void);
```

Get_Sub liefert einen Zeiger auf den Unterknoten mit dem übergebenen Index (aindex) in axml zurück.

```
VIR c_hresult     SC   Get_Sub             (c_dword    aindex,
                                            rp_CXMLBase axml);
```

Add_New_Sub_Node erstellt einen neuen Unterknoten und liefert einen Zeiger auf das Objekt (axml) zurück.

```
VIR c_hresult     SC   Add_New_Sub_Node    (rp_CXMLNode axml);
```

1.2.5.3 CXMLBase *(Klasse)* (abgeleitet von CBase)

CXMLBase ist die grundlegende Klasse des XMLParsers. Hier werden die Prototypen für das prinzipielle Verhalten gelegt.

Members	m_string	p_CString	ein General-Purpose-String
	m_offset	dword	der Offset eines XML-Objekts in seinen übergeordneten Knoten (in Zeichen)

Methoden

FromString erstellt ein XML-Objekt (abgeleitet von CXMLBase) aus einem String (astring). Dabei werden die zu dem Objekt gehörenden Daten aus dem String gelöscht.

```
VIR c_hresult SC FromString    (cp_CString astring);
```

ToString veranlasst ein XML-Objekt, sich in einen String (astring) zu schreiben.

```
VIR c_hresult SC ToString      (cp_CString astring);
```

Clear veranlasst ein XML-Objekt, seine gesamten Daten inclusive seiner Unterobjekte zu löschen.

```
VIR c_hresult SC Clear         (void);
```

Type liefert den genauen Typ eines XML-Objekts als ganzzahligen Code zurück.

XML_BASE	0	(wird hier zurückgegeben)
XML_DOCUMENT	1	
XML_NODE	2	
XML_PROCESSING_INSTRUCTION	4	

```
VIR c_dword       SC   Type                (void);
```

Get_Text liefert den Text eines XML-Objekts in einem String (astring) zurück.

```
VIR c_hresult     SC   Get_Text            (cp_CString astring);
```

Set_Text setzt den Text eines XML-Objekts auf den Wert eines Strings (astring).

```
VIR c_hresult     SC   Set_Text            (cp_CString astring);
```

1.2.5.4 CXMLNode (Klasse) (abgeleitet von CXMLBase)

CXMLNode repräsentiert die normalen XML-Knoten.
Sie treten in der Form <Name Attribut="Wert">Text</Name> bzw. <Name /> auf und können beliebig tief geschachtelt werden.

Members			
	m_string	p_CString	Name eines XML-Knotenobjekts
	m_offset	dword	der Offset eines XML-Objekts in seinen übergeordneten Knoten (in Zeichen)
	m_subnodes	p_CObjectList	Liste der Unterknoten eines XML-Knotenobjekts
	m_attributes	p_CDoubleStringList	Attribute (Name und zugehöriger Wert) eines XML-Knotenobjekts

Methoden

Die Funktion Fromstring überschreibt die bereits existierende Funktion Fromstring aus der Klasse CXMLBase.
Fromstring erstellt ein Knotenobjekt aus einem String (astring). Dabei werden die zu dem Objekt gehörenden Daten aus dem String gelesen und anschließend aus dem String gelöscht.

```
VIR c_hresult     SC   FromString          (cp_CString astring);
```

Die Funktion Tostring überschreibt die bereits existierende Funktion Tostring aus der Klasse CXMLBase.
Tostring veranlasst ein Knotenobjekt, sich in einen String (astring) zu schreiben.

```
VIR c_hresult     SC   ToString            (cp_CString astring);
```

Die Funktion Clear überschreibt die bereits existierende Funktion Clear aus der Klasse CXMLBase.
Clear veranlasst ein Knotenobjekt, seine gesamten Daten inclusive seiner Unterobjekte zu löschen.

```
VIR c_hresult     SC   Clear               (void);
```

Die Funktion Type überschreibt die bereits existierende Funktion Type aus der Klasse CXMLBase.
Type liefert den Code-Wert für ein Knotenobjekt zurück. (hier 2)

```
VIR c_dword       SC   Type                (void);
```

Get_Name liefert den Namen eines XML-Knotenobjekts in einem String (astring) zurück.

```
VIR c_hresult     SC   Get_Name            (cp_CString astring);
```

Set_Name setzt den Namen eines XML-Knotenobjekts auf den Wert eines Strings (astring).

```
VIR c_hresult     SC   Set_Name            (cp_CString astring);
```

Get_Att_Val liefert den zu einem in einem String (aattr) übergebenen Namen eines Attributs zugehörigen Wert des Attributs in einem String (aval) zurück.

```
VIR c_hresult     SC   Get_Att_Val         (cp_CString aattr,
                                            cp_CString aval);
```

Set_Att_Val setzt den zu einem in einem String (aattr) übergebenen Namen eines Attributs zugehörigen Wert des Attributs auf den Wert eines Strings (aval). Falls der Attributname noch nicht vorhanden ist, wird das Attribut (Name und Wert) zu den bisherigen hinzugefügt.

```
VIR c_hresult     SC   Set_Att_Val         (cp_CString aattr,
                                            cp_CString aval);
```

Get_Sub_Count liefert die Anzahl der Kindknoten eines Knotenobjekts zurück.

```
VIR c_dword       SC   Get_Sub_Count       (void);
```

Get_Sub liefert einen Zeiger auf den Unterknoten mit dem übergebenen Index (aindex) in axml

zurück.

```
VIR c_hresult      SC   Get_Sub              (c_dword    aindex,
                                               rp_CXMLBase axml);
```

Add_New_Sub_Node erstellt einen neuen Unterknoten und liefert einen Zeiger auf das Objekt (axml) zurück.

```
VIR c_hresult      SC   Add_New_Sub_Node      (rp_CXMLNode axml);
```

1.2.5.5 CXMLPI (Klasse) (abgeleitet von CXMLBase)

CXMLPI repräsentiert eine XML-Prozessanweisungsobjekt, welches lediglich über einen Text verfügt. Sie treten in der Form <? text ?> auf.

Methoden

Die Funktion Fromstring überschreibt die bereits existierende Funktion Fromstring aus der Klasse CXMLBase, wenn es sich um ein XML-Prozessanweisungsobjekt handelt. Fromstring erstellt ein Prozessanweisungsobjekt aus einem String (astring). Dabei werden die zu dem Objekt gehörenden Daten aus dem String gelöscht.

```
VIR c_hresult      SC   FromString           (cp_CString astring);
```

Die Funktion Tostring überschreibt die bereits existierende Funktion Tostring aus der Klasse CXMLBase, wenn es sich um ein XML-Prozessanweisungsobjekt handelt. Tostring veranlasst ein Prozessanweisungsobjekt, sich in einen String (astring) zu schreiben.

```
VIR c_hresult      SC   ToString             (cp_CString astring);
```

Die Funktion Type überschreibt die bereits existierende Funktion Type aus der Klasse CXMLBase, wenn es sich um ein XML-Prozessanweisungsobjekt handelt. Type liefert den Code-Wert für ein Prozessanweisungsobjekt zurück.

```
VIR c_dword        SC   Type                 (void);
```

1.2.5.5 xml_parse *(Funktion)*

```
API c_hresult SC XML_Parse  (p_CString astring, rp_CXML axml_tree);
```

xml_parse liefert ein XML-Objekt (axml_tree), welches aus einem String (astring) erstellt wurde.

1.2.6 Modul MathServer

Modul:	MathServer	**Bearbeiter:**	Thomas Weise
Bezeichner:	MathServer	**Änderungsstand:**	12.06.2003

Kurzbeschreibung:

Das Modul MathServer bindet das Modul Engine ein, welches genutzt wird um den Server-Thread zu starten. Es exportiert die gesamte Serverfunktionalität über die TCP/IP-Schnittstelle.

Export:	Server-Funktionalität	**Import:**	Engine

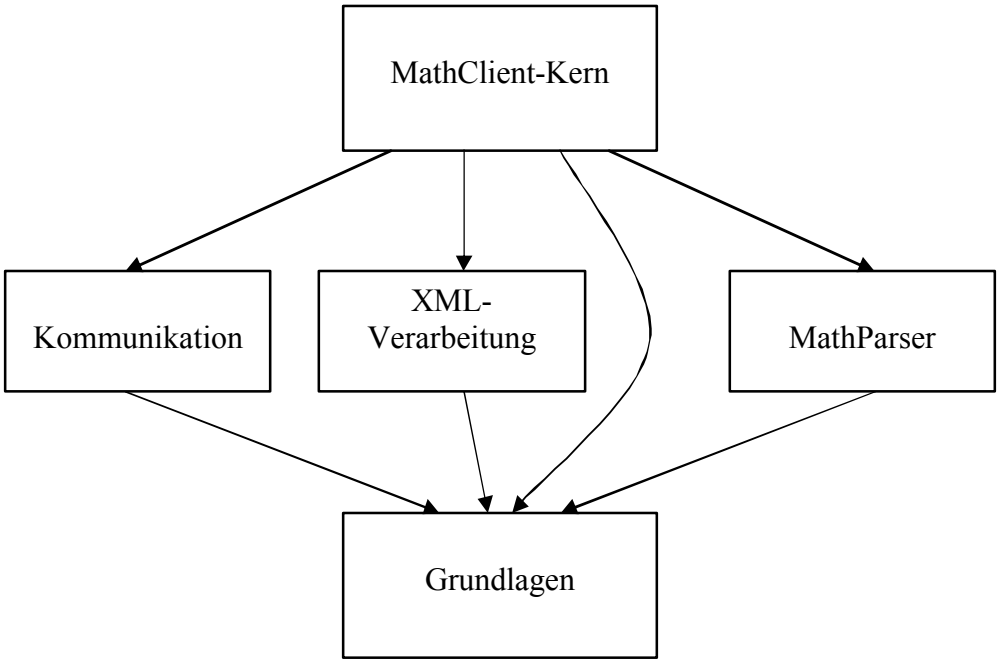
1.2.7 Modul MathClient

Modul:	MathClient	**Bearbeiter:**	Rico Roßberg
Bezeichner:	MathClient	**Änderungsstand:**	16.06.2003

Kurzbeschreibung:

Das Modul MathClient enthält das Anwenderinterface des Systems, und zerfällt in Untermodule, die jeweils Teilfunktionen des MathClient realisieren.

Export:	-	**Import:**	MathServer



1.2.7.1 Untermodul Grundlagen

Das Untermodul Grundlagen enthält Grundfunktionen, die den anderen MathClient-Modulen zur Verfügung stehen.

Export: MTreeNode
PrintText
DrawTree
Weitere allgemeine Grundfunktionen, auf die hier nicht weiter eingegangen wird

Import: -

1.2.7.1.1 MTreeNode (Klasse)

Diese Klasse repräsentiert einen Knoten des Baumes Eingabetext.

Eigenschaften:			
	Data	String	Die Daten des Knotens
	ID	Long	Die ID des Knotens
	Children	Variant	Dynamisches Feld aus MTreeNode
	NumberOfChildren	Long	Anzahl der Elemente von Children

1.2.7.1.2 PrintText (Prozedur)

Diese Prozedur dient zum Drucken ausgewählter Daten.

Sub PrintText(PrintData as String)

1.2.7.1.3 DrawTree (Prozedur)

Diese Prozedur stellt einen mathematischen Ausdruck grafisch dar

Sub DrawTree(MTree as MTreeNode, DrawBox as PictureBox, Optional MarkedPartID as Long)

1.2.7.2 Untermodul Kommunikation

Dieses Modul realisiert die Kommunikation des MathClient mit dem MathServer, unter Verwendung der TCP/IP-Schnittstelle.

Export: TalkToServer

Import: Grundlagen

1.2.7.2.1 TalkToServer (Funktion)

Sendet die Anfrage vom MathClient an den MathServer, und empfängt dessen Antwort.

Function TalkToServer(SendData as String, ServerIP as Long, ServerPort as Long) as String

1.2.7.3 Untermodul XML-Verarbeitung

Dieses Modul erzeugt den an den Server zu sendenden XML-String, und verarbeitet dessen Antwort.

Export: TreeToXMLString
FindSpecificSolution

Import: Grundlagen

1.2.7.3.1 TreeToXMLString (Funktion)

Wandelt den vom MathParser erzeugten Baum in einen XML-String um

Function TreeToXMLString(MTreeRoot as MTreeNode) as String

1.2.7.3.2 FindSpecificSolution (Funktion)

Findet in der Serverantwort ein bestimmtes Zwischenergebnis anhand der ID.

Function FindSpecificSolution(SearchString as String, ID as Long) as String

1.2.7.4 Untermodul MathParser

Wandelt den vom Nutzer eingegebenen mathematischen Ausdruck in eine XML-Baumstruktur um

Export: MathStringToTree

Import: Grundlagen

1.2.7.4.1 MathStringToTree (Funktion)

Wandelt den vom Nutzer eingegebenen mathematischen Ausdruck in eine XML-Baumstruktur um

Function MathStringToTree(MathString as String) as MTreeNode

1.2.7.5 Untermodul MathClient-Kern

Verknüpft die Funktionalität der untergeordneten Module zur Gesamtfunktion des MathClient, und beinhaltet die eigentliche Nutzeroberfläche.

Export: -

Import: Alle untergeordneten Module

1.2.8 TCP/IP-Schnittstelle

Kurzbeschreibung:

Die TCP/IP-Schnittstelle dient dem Versenden der Daten zwischen Klienten und Server. Sie ist bidirektional und basiert auf den Microsoft Windows Socket bzw. HTTP-Spezifikationen.

Wie bereits in Teilbeleg 1 erwähnt wird ein HTTP-kompatibles Protokoll verwendet. Da es nur der Datenübertragung dient, beschränken wir uns darauf, den Befehl „GET“ zu unterstützen. Anstelle des URL-Teils wird das Wort „solution“ eingetragen.
Hierbei kann die Anzahl der gewünschten Vorkommastellen (decimals) und Nachkommastellen (fractals) definiert werden.
Das Startsymbol für den gesamten Traffic ist <traffic>, für die Datenübertragung Klient-Server <anfrage> und Server-Klient <antwort>.

Tabelle 1.2.8.1

Element	Strukturbeschreibung
<http-befehl>	{"GET"}
<typ>	{"solution decimals=" <zfolge> “ fractals=“ <zfolge> }
<LF>	Zeilenumbruch
<leerzeile>	<LF><LF>
<xml-daten>	XML-Text für Anfrage- und Antwortdaten
<ergebnis-nr>	{"200 OK" | "8" <ziffer><ziffer> " Fehler"}
<anfrage>	{<http-befehl>" "<typ>" HTTP/" <ziffer> "." <ziffer> <leerzeile> <xml-daten>}
<antwort>	{"HTTP/" <ziffer> "." <ziffer> " " <ergebnis-nr> <LF> “Content-Type: text/mathml+xml“ <leerzeile> <xml-answer>}
<traffic>	{<anfrage> | <antwort>}

Die XML-Daten der Antwort (entspricht dem Symbol <xml-daten> der <antwort> des Servers) erfolgt nach der MathML-Syntax.
Die einzelnen Ergebnisse (Endergebnis und Zwischenergebnisse) werden je in <cn>-tags (<cn> </cn>) geklammert.
Der <cn>-tag, als einiger zulässiger Datenübergabe-Tag, erfährt eine id-Nummer als Attribut <cn id = [ganze positive Zahl]>.
Die id-nummer stellt die Verknüpfung zwischen dem jeweiligen Ergebnis und dem zugehörigen Berechnungsabschnitt dar.

Tabelle 1.2.8.2

Element	Strukturbeschreibung
<xml-pi>	XML Processing-Instructions
<xml-dt>	XML Comment
<xml-mathml>	XML Comment
<xml-math>	“<math>“ <xml-cnc> “</math>“
<xml-cn>	“<cn id=‘“ <zfolge> “’>“ <zahl> “</cn>“
<xml-cnc>	<xml-cn> | <xml-cn> <xml-cnc>
<xml-answer>	<xml-pi> <xml-dt> <xml-mathml>

Softwarepraktikum 2003

MathServer

Projektdokumentation
(Teilbeleg 3)

Teamleiter: Thomas Weise
Mitglieder des Projektteams:
Roland Fischer
René Kreiner
Rico Roßberg
Thomas Ziegs

Praktikumsbetreuer : Dipl.-Inf. Lutz Neugebauer

Chemnitz, den 09.07.2003

1 Programmdokumentation

1.1 Definition der Modulinterfaces

1.1.1 MathServer

Die Module des MathServers sind in VisualBasic programmiert.

1.1.1.1 Modul Basics

Zuständiger: Thomas Weise

Das Modul Basics stellt als Grundlage aller anderen Module atomare Funktionen und Klassen zu Verfügung. Es importiert alle Windows-Routinen die im gesamten MathServer verwendet werden. Es werden keinerlei C-native Routinen angewandt, um maximale Performance zu gewährleisten basiert sämtliche Programmierarbeit auf der Windows API (Application Programming Interface) und selbst entwickeltem Code, vorzugsweise Assembler.

In der Modulbeschreibung zum Modul Basics soll nur näher auf die funktions- oder technikintegralen Bestandteile näher eingegangen werden. Weniger bestimmende Faktoren wie z.B. Listen wurden bereits in Teilbeleg 2 ausreichend abgehandelt, sie sind unter ihrer Definition auch eher trivial ausgelegt, es sind eben Listen, und daher hier nicht von Interesse. Auch würde z.B. der Quelltext von CString.cpp (ohne Kommentare) alleine bereits ca. 50 Seiten füllen.

1.1.1.1.1 Header CBase.h

CBase ist die wichtigste Klasse des rojekts, alle anderen leiten sich von ihr ab. Ihre Schnittstelle wird im Header CBase.h definiert.

```
#ifndef         CBASE_USED
#define         CBASE_USED

#include        "..\BASIC_DEFINITIONS.h"
#include        "..\BASIC_TYPES.h"
#include        "..\BASIC_ROUTINES.h"
#define         DEATH_STATE             DWORD_MSB
#define         FORBIDDEN_STATE         DEATH_STATE

CLASSAPI                                        CBase;

// see basic_macros for that
DEFINE_TYPES(CBase)

CLASSAPI                                        CBase

  {
  public:
//Konstruktor, athreadsecured hat in dieser Anwendung keine Bedeutung
                        CBase               (c_boolean athreadsecured = false);
//Destruktor
  VIR                   ~CBase              (void);

//jede abgeleitete Klasse kann die 32-Bit Variable für den Status verwenden und mitState, GetState und SetState
//manipulieren. Reserviert ist der DEATH_STATE, das Bit 31. Es wird verwendet, um einen Destruktorlauf zu
//verhindern
  VIR c_dword       SC    State               (void);

//Gibt wieder, ob eine bestimmte Statusbitkombination gesetzt ist
  VIR c_boolean     SC    GetState            (c_dword astate);
```

```
protected:

//SetState soll nur von Methode der Klasse selbst aufgerufen werden
  VIR c_dword       SC   SetState             (c_dword astate, c_boolean aset);

public:

//gibt die aktuelle Zahl von Referenzen auf das Objekt zurück
  VIR c_dword       SC   RefCount             (void);

//erhöht die Zahl der Referenzen um 1, muss aufgerufen werden, wann immer ein neuer Zeiger auf das Objektt
//zeigen soll
  VIR c_dword       SC   AddRef               (void);

//erniedrigt den Referenzzähler um 1, gibt das Objekt frei, wenn er 0 erreicht
//muss immer dann aufgerufen werden, wenn ein Zeiger vom Objekt weggeboge wird
  VIR c_dword       SC   Release              (void);


//die Methoden Enter und Leave haben in dieser Anwendung keine direkte Bedeutung,
//sie wurden vorsorglich entwickelt, als die genauen Rahmenbedingungen der
//Anwendungen noch nicht voll spezifiziert waren, ermöglichen jedoch evtl. späteres
//Erweitern, so dass mehrere MathServer gleichzeitig an einer Aufgabe rechnen
//können
  VIR void          SC   Enter                (void);
  VIR void          SC   Leave                (void);

//die operatoren new und delete werden überschrieben um die C++-Wrapper für die
//Windows API zu umgehen, und eigene verwenden zu können
      c_pointer           operator new         (unsigned int asize);
      void                operator delete      (void* amemory);


  private:

                dword   m_state;   //32-Bit Statusvariable
                dword   m_refcount; //Referenzzähler
                dword   m_mutex;  //eine Mutex, um das Objekt evtl. zu sperren, siehe Enter, Leave

  };


#define           CBASE_VTBL_LEN              9

#endif
```

1.1.1.1.2 Header CSocket.h

Im Sockets-Modul wird die Netzwerkfähigkeit der Applikation implementiert.

```
#ifndef          CSOCKET_USED
#define          CSOCKET_USED

#include         "..\BASIC_DEFINITIONS.h"
#include         "..\BASIC_TYPES.h"
#include         "..\BASIC_ROUTINES.h"
#include         "CSTREAM.h"
#include         "CSTRING.h"



#define          CSOCKET_SERVER    1
#define          CSOCKET_CLIENT    2
#define          CSOCKET_ALL       (CSOCKET_SERVER | CSOCKET_CLIENT)
#define          CSOCKET_MAX       CSOCKET_CLIENT

CLASSAPI                             CSocket;


DEFINE_TYPES(CSocket)


CLASSAPI                             CSocket : public CStream

  {

  public:

//Konstruktor und Destruktor
                          CSocket             (c_boolean athreadsecured = false);
  VIR                    ~CSocket             (void);

//gibt wieder, ob ein Socket geöffnet ist
  VIR c_boolean      SC    IsOpen              (void);

//gibt wieder, ob von einem Socket gelesen werden kann, gelesen kann nur von Klienten werden
  VIR c_boolean      SC    CanRead             (void);

//gibt wieder, ob in ein Socket geschrieben werden kann, gelesen kann nur von Klienten werden
  VIR c_boolean      SC    CanWrite            (void);

//beendet eine Internetverbindung
  VIR c_hresult      SC    Close               (void);

//teilt dem Socket mit, dass es als Server an ein bestimmte Adresse/Port gebunden werden soll
  VIR c_hresult      SC    Server              (p_CString abindaddress);

//Akzeptiert eine einkommende Verbindung, schafft sie auf ein neues Klientensocket
//geht nur bei Server-Sockets
//aterminateproc wird als Callback-Routine regelmäßig mit aparam aufgerufen, wenn es gesetzt ist,
//um festzustellen, ob der Server beendet werden soll
  VIR c_hresult      SC    Accept              (rp_CSocket anewsock,
                                                c_pointer  aterminateproc = 0,
                                                c_dword    aparam         = 0);

//verbindet sich als Klient mit dem Server, welcher an einer spezifischen Adresse wartet
  VIR c_hresult      SC    Client              (p_CString aconnectaddress);

//schreibt einen Buffer über die Netzwerkverbindung, kann nur von Klientensockets durchgeführt werden
  VIR c_hresult      SC    Write               (c_pointer adata, c_dword asize,
                                                r_dword awritten);
```

```
//ließt einen Buffer aus dem Netzwerk, kann evtl. weniger lesen, als angefordert, geht nur bei Klientensockets

VIR c_hresult     SC   Read                (c_pointer adata, c_dword asize,
                                             r_dword   aread);

//gibt die aktuelle Adresse des Sockets aus
  VIR c_hresult     SC   Address             (p_CString aaddress);

  private:
                dword   m_socket; //das Sockethandle
          sockaddr_in   m_address; //die Socketadresse
                dword   m_event; //ein Event, zu Synchronisation

  };

c_hresult     FC   StartWSA(void);
c_hresult     FC   CloseWSA(void);

#endif
```

1.1.1.1.3 Header CThread.h

Die Klasse CThread aus diesem Modul bildet sozusagen das Rückgrat der Engine.

```
#ifndef         CTHREAD_USED
#define         CTHREAD_USED

#include        "..\BASIC_DEFINITIONS.h"
#include        "..\BASIC_TYPES.h"
#include        "..\BASIC_ROUTINES.h"
#include        "CObjectList.h"


CLASSAPI                            CThread;

DEFINE_TYPES(CThread)

#define   CTHREAD_RUNNING     (1)
#define   CTHREAD_TERMINATED  (CTHREAD_RUNNING * 2)
#define   CTHREAD_SUSPENDED   (CTHREAD_RUNNING * 4)
#define   CTHREAD_FIRST       (CTHREAD_RUNNING * 8)

CLASSAPI                            CThread : public CObjectList

  {

  public:

//Als owner muss der aufrufende Thread angegeben werden. Wird ein Thread im Hauptprogramm gestartet
//so wird aowner auf 0 gesetzt.
                        CThread             (p_CThread aowner = 0);
  VIR                   ~CThread            (void);

  protected:

//Mainthread dient als Framwork für Execute und wird als Threadroutine gestartet
  VIR c_hresult     SC    MainThread          (void);                       // 001

  public:
//Execute wird für die spezifischen Aufgaben der abgeleiteten Threads überschrieben
  VIR c_hresult     SC    Execute             (void);                       // 002

//Startet einen Thread (auch aus Ruhezustand)
  VIR c_hresult     SC    Resume              (void);                       // 003

//Versetzt einen Thread in den Ruhezustand.
  VIR c_hresult     SC    Suspend             (void);                       // 004

//Teilt dem Thread mit, dass er seine Ausführung an der nächstmöglichen, sicheren Stelle beenden soll.
  VIR c_hresult     SC    Terminate           (void);                       // 005

  protected:
//Einige Methoden, um Subthreads zu terminieren oder auf sie zu warten.
  VIR c_hresult     SC    TerminateSubThreads (void);                       // 006
  VIR c_hresult     SC    WaitForSubThreads   (c_dword atime = INFINITE);   // 007
  VIR c_hresult     SC    WaitForMainThread   (c_dword atime = INFINITE);   // 008

  public:
//Wartet auf Beedigung des Threads
  VIR c_hresult     SC    WaitFor             (c_dword atime = INFINITE);   // 009
//Teilt dem Thread mit, dass er seine Ausführung an der nächst sicheren Stelle beenden soll, und wartet darauf
  VIR c_hresult     SC    TerminateAndWait    (void);                       // 010

//gibt den Exitcode des Threads zurück
  VIR c_hresult     SC    ExitCode            (void);                       // 011
```

```
//gibt den systemweit eindeutigen Identifier des Threads zurück
  VIR c_dword       SC   Id                  (void);                  // 012

//setzt die Priorität des Threads
  VIR c_hresult     SC   SetPriority         (c_dword apriority);     // 013

//.. und gibt sie zurück
  VIR c_dword       SC   GetPriority         (void);                  // 014

//Teilt mit, ob der Thread bereits jemals gestartet wurde
  VIR c_boolean     SC   Started             (void);                  // 015

//Teilt mit, ob er gerade ausgeführt wird.
  VIR c_boolean     SC   Running             (void);                  // 016

//..oder ob er sich im Ruhezustand befindet.
  VIR c_boolean     SC   Suspended           (void);                  // 017

//Teilt mit, ob er beendet wurde
  VIR c_boolean     SC   Terminated          (void);                  // 018

//übergibt die Programmkontrolle an einen anderen Thread.
  VIR void          SC   SwitchToOther       (void);                  // 019

  protected:

//interne Routinen zum Subthread management
  VIR c_hresult     SC   SetLength           (c_dword alength);
  VIR c_hresult     SC   Delete              (c_dword aindex);
  VIR c_hresult     SC   Remove              (c_pointer aitem);


  private:

              dword   m_handle;   //handle des Threads
              dword   m_id; //id des Threads
              dword   m_subthreadsevent; //Event für die Subthreads
          p_CThread   m_owner; //übergeordneter Thread
            hresult   m_exitcode; //Rückgabewert

  };


#define     CTHREAD_VTBL_LEN          ((COBJECTLIST_VTBL_LEN) + 19)
#define     CTHREAD_TERMINATED_INDEX  ((COBJECTLIST_VTBL_LEN) + 17)


#endif
```

1.1.1.2 Modul Engine

Zuständiger: Thomas Weise

1.1.1.2.1 Header Engine_Processor.h

Der Engine-Prozessor bearbeitet je eine Serveranfrage. Er stellt einen Thread, einen einzelnen Programmstrang dar, welcher gleichläufig mit allen anderen Strängen ausgeführt wird. Siehe auch CThread. Zu beachten ist. dass er je einen Eingangsstream und einen Ausgangsstream erhält. Im Projekt wird das je ein und das selbe Socket für die Netzwerkverbindung sein. Man könnte jedoch auch, ohne eine einzige Zeile Kode ändern zu müssen, die Anfrage aus einer Datei lesen und die Antwort an einen Netzwerkklienten schicken lassen.

```
#ifndef         ENGINE_PROCESSOR_USED
#define         ENGINE_PROCESSOR_USED

CLASSAPI                            CProcessor;

DEFINE_TYPES(CProcessor)

CLASSAPI                            CProcessor : public CThread

  {

  public:

                        CProcessor          (p_CStream aindata,
                                             p_CStream aoutdata,
                                             p_CThread aowner = 0);
  VIR                   ~CProcessor         (void);

//die überschriebene Execute-Methode enthält den eigentlichen Kode
  VIR c_hresult     SC  Execute             (void);

  private:

          p_CHTTPStream   m_indata; //Eingangsdaten
          p_CHTTPStream   m_outdata; //Ausgangsdaten

  };


#endif
```

1.1.1.2.2 Header Engine_Server.h

Die Klasse CServer stellt je eine Instanz des Servers dar. Sie bekommt einen Port mitgeteilt, an dem sie auf eingehende Verbindungen warten soll. Die Routinen für Verbindungsannahme laufen wieder in einem Thread, so dass es auch ohne weiteres möglich ist, dass das Hauptprogramm selbst gleichzeitig als Server (mit eventuell verschiedenen Konfigurationen) an mehreren Ports „listened“.

```
// here come the server routines

#ifndef          ENGINE_SERVER_USED
#define          ENGINE_SERVER_USED

#include         "Engine_Processor.h"


CLASSAPI                                        CServer;


DEFINE_TYPES(CServer)

CLASSAPI                                        CServer : public CThread

  {

  public:


                          CServer               (p_CString aport);
  VIR                     ~CServer              (void);

//die überschriebene Execute-Methode enthält den eigentlichen Kode

  VIR c_hresult     SC    Execute               (void);

  private:

             p_CSocket    m_socket; //serversocket


  };

#endif
```

1.1.1.3 Modul Formelparser

Zuständiger: Rico Roßberg

1.1.1.3.1 Header Formel.h

Im Formelparser (der nur den Header Formel hat) werden die mathematischen Routinen und der XML-Parser verwendet, um die empfangenen Eingangsdaten (MathML) in Ausgangsdaten (wieder MathML) umzurechnen.

```
#ifndef          FORMEL_USED
#define          FORMEL_USED



#include         "..\BASICS.h"
#include         "..\Math\Math.h"
#include         "..\XML_Parser\XML.h"

#ifdef           FORMEL_PARSER_DLL_EXPORTS

#undef           API
#undef           CLASSAPI
#define          API                 EXP
#define          CLASSAPI            CLASSEXP

#else

#undef           API
#undef           CLASSAPI
#define          API                 IMP
#define          CLASSAPI            CLASSIMP

#endif


//Diese Funktione parst einen Formel, der Daten sich in XML-Format in astring befinden
//und schreibt das Ergebnis in aresult.
//Die zu verwendente Anzahl von Vorkomma- und Nachkommastellen steht jeweils in
//adecimals und afractals
API c_hresult    SC  Formula_Parse       (p_CString astring, p_CString aresult,
                                          c_dword   adecimals, c_dword afractals);


#endif
```

1.1.1.4 Modul Math

Zuständiger: Thomas Ziegs

1.1.1.4.1 Header RealFunctions.h

Hier werden die mathematischen Routinen definiert, die vom Wrapper CReal dann umschlossen werden.
Alength: Länge aller Pointerzahlen.

```
API c_pointer   FC    _NEW          (c_dword        alength)
```
Gibt Pointer zurück, der auf den neuen Speicher für eine Zahl zeigt die die Länge alength hat.

```
API c_pointer   FC    _COPY         (c_dword        alength,
                                     c_pointer      aoriginal)
```
Kopiert Pointer aoriginal mit Länge alength auf den Rückgabewert.

```
API void        FC    _COPYBITS     (c_dword        alength,
                                     c_pointer      aoriginal,
                                     c_dword        aleft,
                                     c_dword        aright,
                                     c_pointer      aoutdata)
```
Kopiert bitweise aoriginal zu aoutdata, von Position aleft bis arigth. Alength: Länge der Zahl.

```
API void        FC    _FREE         (r_pointer      anum)
```
Gibt den Speicher des Pointers anum frei.

```
API void        FC    _NOT          (c_dword        length,
                                     c_pointer      aindata,
                                     c_pointer      aoutdata)
```
Bildet das 1er-Komplement von aindata und schreibt in aoutdata. Alength: Länge der Zahl.

```
API void        FC    _NOT2         (c_dword        alength,
                                     c_pointer      adata,
                                     c_pointer      aoutdata)
```
Bildet das 2er-Komplement von adata und schreibt in aoutdata. Alength: Länge der Zahl.

```
API c_boolean   FC    _POSITIVE     (c_dword        alength,
                                     c_pointer      aindata)
```
Gibt true zurück wenn das MSB von aindata mit Länge alength 0 ist.

```
API c_boolean   FC    _NEGATIVE     (c_dword        alength,
                                     c_pointer      aindata)
```
Gibt true zurück wenn das MSB von aindata mit Länge alength 1 ist.

```
API c_boolean   FC    _TEST_BIT     (c_dword        alength,
                                     c_dword        abit,
                                     c_pointer      adata)
```
Testet an Position abit in adata mit Länge alength auf 1 und iefert true wenn 1.

```
API void        FC    _INS_BIT      (c_dword        alength,
                                     c_dword        abitnum,
                                     c_boolean      aooz,
                                     c_pointer      adata)
```
Setzt Bitwert aooz an Position abitnum in adata mit Länge alength.

```
API c_dword     FC    _MSB          (c_dword        alength,
                                     c_pointer      adata)
```
Gibt als c_dword den Index des MSB von adata zurück. Alength: Länge der Zahl.

```
API c_dword     FC    _LSB          (c_dword        alength,
                                     c_pointer      adata)
```
Gibt als c_dword den Index des LSB von adata zurück. Alength: Länge der Zahl.

```
API c_dword     FC    _CMP          (c_dword        alength,
                                     c_pointer      aindata1,
                                     c_pointer      aindata2)
```
Vergleicht aindata1 mit aindata2 und gibt in als c_dword Ergebnis zurück. Alength: Länge der Zahl.

```
API c_boolean   FC    _IZ           (c_dword        alength,
                                     c_pointer      aindata)
```
Liefert true wenn aindata=0. Alength: Länge der Zahl.

```
API c_boolean   FC    _SHL_1        (c_dword        alength,
                                     c_pointer      aindata,
                                     c_pointer      aoutdata)
```
Linksschieben von aindata um 1 Stelle, Speicherung in aoutdata. Alength: Länge der Zahl.

```
__forceinline c_boolean   FC    _SHL_N_MOD32    (c_dword        alength,
                                                 c_dword        adigits,
                                                 c_pointer      aindata,
                                                 c_pointer      aoutdata)
```
Linksschieben von aindata um adigits Stellen (<32), Speicherung in aotdata. ALength: Länge der Zahl.

```
__forceinline  c_boolean   FC   _SHL_N_32       (c_dword        alength,
                                                 c_dword        adigits,
                                                 c_pointer      aindata,
                                                 c_pointer      aoutdata)
```
Linksschieben von aindata um adigits 32Bit-Blöcke, Speicherung in aoutdata. Alength: Länge der Zahl.

```
API c_boolean   FC    _SHL_N        (c_dword        alength,
                                     c_dword        adigits,
                                     c_pointer      aindata,
                                     c_pointer      aoutdata)
```
Schiebt aindata um adigits Stellen nach links, Ergebnis aoutdata wird gebildet. Alength: Länge der Zahl.

```
API c_boolean   FC    _SHR_1        (c_dword        alength,
                                     c_pointer      aindata,
                                     c_pointer      aoutdata)
```
Rechtsschieben von aindata um 1 Stelle, Speicherung in aoutdata. Alength: Länge der Zahl.

```
__forceinline c_boolean   FC    _SHR_N_MOD32    (c_dword        alength,
                                                 c_dword        adigits,
                                                 c_pointer      aindata,
                                                 c_pointer      aoutdata)
```
Rechtsschieben von aindata um adigits Stellen (<32), Speicherung in aotdata. ALength: Länge der Zahl.

```
__forceinline c_boolean   FC    _SHR_N_32       (c_dword        alength,
                                                 c_dword        adigits,
                                                 c_pointer      aindata,
                                                 c_pointer      aoutdata)
```
Rechtsschieben von aindata um adigits 32Bit-Blöcke, Speicherung in aoutdata. Alength: Länge der Zahl.

```
API c_boolean   FC    _SHR_N        (c_dword        alength,
                                     c_dword        adigits,
                                     c_pointer      aindata,
                                     c_pointer      aoutdata)
```
Schiebt aindata um adigits Stellen nach rechts, Ergebnis aoutdata wird gebildet. Alength: Länge der Zahl.

```
API c_boolean   FC    _ADD          (c_dword        alength,
                                     c_pointer      aindata1,
                                     c_pointer      aindata2,
                                     c_pointer      aoutdata)
```

Addiert aindata1 mit aindata2. Alength: Länge der Zahl; aindata1: Operand 1; aindata2: Operand2; aoutdata: Ergebnis.

```
API c_boolean   FC    _SUB          (c_dword        alength,
                                     c_pointer      aindata1,
                                     c_pointer      aindata2,
                                     c_pointer      aoutdata)
```

Subtrahiert aindata2 von aindata1. Alength: Länge der Zahl; adot: Kommaposition in der Zahl; aindata1: Operand 1; aindata2: Operand2; aoutdata: Ergebnis.

```
API c_boolean   FC    _MUL          (c_dword         alength,
                                     c_dword         adot,
                                     c_pointer       aindata1,
                                     c_pointer       aindata2,
                                     c_pointer       aoutdata,
                                     r_boolean       aunderflow)
```

Multipliziert aindata1 mit aindata2. Alength: Länge der Zahl; adot: Kommaposition in der Zahl; aindata1: Operand 1; aindata2: Operand2; aoutdata: Ergebnis; aunderflow: Unterlauf.

```
API c_boolean   FC    _DIV          (c_dword         alength,
                                     c_dword         adot,
                                     c_pointer       aindata1,
                                     c_pointer       aindata2,
                                     c_pointer       aoutdata,
                                     r_boolean       aunderflow,
                                     r_boolean       div0)
```

Dividiert aindata1 durch aindata2. Alength: Länge der Zahl; adot: Kommaposition; aindata1, aindata2 sind Operanden; aoutdata: Ergebnis; aunderflow: wenn bei Berechnung Unterlauf entsteht; div0 bei Division durch Null. Funktioniert nach dem Shift & Subverfahren.

```
__forceinline c_boolean   FC    _STR_MUL_10        (c_dword         alength,
                                                    c_pointer       aindata,
                                                    c_pointer       aoutdata,
                                                    r_boolean       aunderflow)
```

Multipliziert Zahl mit 10 vorzeichenlos. Rechnet nach gleichem Prinzip wie _MUL (Shift & Add) wobei Shift & Add direkt mit seperaten Befehlen realisiert wird.

```
__forceinline c_boolean   FC    _ADD_DWORD         (c_dword         alength,
                                                    c_pointer       aindata1,
                                                    c_dword         aindata2,
                                                    c_pointer       aoutdata)
```

Addiert dword-Zahl (aindata2) zu Pointer-Zahl (aindata1) indem dword wie ein Pointer gebraucht wird unter Beachtung der dwordgrenzen. Stellenweise Addition mit Übertrag. Alength: Länge unserer Zahlen; aoutdata: Ergebnis.

```
API c_hresult   FC    _STR2NUM      (c_string       astring,
                                     c_dword        astringlength,
                                     c_character    astringdot,
                                     c_dword        anumlength,
                                     c_dword        anumdotpos,
                                     c_pointer      anumnew,
                                     r_boolean      aoverflow,
                                     r_boolean      aunderflow)
```

Wandelt String zur Zahl. A_string: der String; astringlength: Stringlänge; astringdot: “.” oder “,”; anumlengt: die Länge der Zahl; anumdotpos: Kommaposition in Zahl; anumnew: die Zahl die wir erstellen wollen; aoverflow: Überlauf; aunderflow: Unterlauf.

```
API void        FC    _INT            (c_dword        alength,
                                       c_dword        adotpos,
                                       c_pointer      aoriginal,
                                       c_pointer      avorkomma)
```

Liefert die Vorkommastellen einer Zahl. Vorkommastellen aoriginal werden nach avorkomma kopiert. A_length: Länge der Zahlen; adotpos: Kommaposition in Zahl; aoriginal: Originalzahl; avorkomma: Vorkommastellenzahl.

```
API void        FC    _FRAC           (c_dword        alength,
                                       c_dword        adotpos,
                                       c_pointer      aoriginal,
                                       c_pointer      anachkomma)
```

Liefert die Nachkommastellen einer Zahl. Nachkommastellen aoriginal werden nach anachkomma kopiert. A_length: Länge der Zahlen; adotpos: Kommaposition in Zahl; aoriginal: Originalzahl; avorkomma: Nachkommastellenzahl.

```
API c_boolean   FC    _I_DIV_MOD      (c_dword        alength,
                                       c_dword        adot,
                                       c_pointer      aindata1,
                                       c_pointer      aindata2,
                                       c_pointer      adiv,
                                       c_pointer      amod,
                                       r_boolean      div0)
```

Funktioniert nach dem gleichem Prinzip wie Division. Allerdings werden keine Nachkommastellen gebildet und ein Rest wird zurückgegeben. A_length: Länge der Zahl; adot: Kommaposition der Zahl; aindata1, aindata2: Zahlen (Operanden); adiv: ganzahliges Divisionsergebnis; amod: Rest; div0: bei durch Null.

```
API c_hresult   FC    _NUM2STR        (c_pointer      anum,
                                       c_dword        anumlength,
                                       c_dword        anumdotpos,
                                       c_character    astringdot,
                                       p_CString      astring,
                                       c_dword        astringkommastellen)
```

Wandelt eine Zahl zum String. Anum: die Zahl; anumlength: Länge der Zahl; anumdotpos: Kommaposition in Zahl; astringdot: Kommazeichen des Strings; astring: zu bildender String, astringkommastellen: Nachkommastellenanz. des Strings.

```
API c_boolean   FC    _IS_MAX_NEG     (c_dword        alength,
                                       c_pointer      aoriginal)
```

Gibt true zurück wenn Zahl min. neg. Zahl.

```
API c_boolean   FC    _ABS            (c_dword        alength,
                                       c_pointer      aoriginal,
                                       c_pointer      aabs)
```

Bildet den Betrag einer Zahl. Betrag der kleinsten neg. Zahl nicht möglich, da 2er Komplement.

```
__forceinline boolean     FC    _INC          (c_dword        alength,
                                               c_pointer      ain,
                                               c_pointer      aout)
```

Addiert auf ain, mit Länge alength (ohne Kommastellen, LSB=0) eine 1. Ergebnis aout.

```
__forceinline boolean     FC    _DEC          (c_dword        alength,
                                               c_pointer      ain,
                                               c_pointer      aout,
                                               r_boolean      aiszero)
```

Subtrahiert von ain, mit Länge alength (ohne Kommastellen, LSB=0) eine 1. Ergebnis aout. Aiszero wird true wenn Ergebnis 0 wird.

```
API c_boolean   FC    _FACTORIAL      (c_dword        alength,
                                       c_dword        adotpos,
                                       c_pointer      ain,
                                       c_pointer      aout)
```

bildet Fakultät einer Zahl

```
API c_boolean    FC     _E_X            (c_dword         alength,
                                         c_dword         adotpos,
                                         c_pointer       ax,
                                         c_pointer       aout)
```

Liefert e^x durch die Formel :

$$e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

Alength: Länge der Zahl ; adotpos: Kommaposition in der Zahl ; ax: Exponent von e; aout: Ergebnis.

```
API c_boolean    FC     _SIN            (c_dword         alength,
                                         c_dword         adotpos,
                                         c_pointer       ain,
                                         c_pointer       aout)
```

Berechnet Sinus einer Zahl mit folgender Formel :

$$\sin x = \sum_{i=0}^{\infty} \frac{x^{2i+1}}{(2i+1)!} * (-1)^i$$

Alength: Länge der Zahl; adotpos: Kommaposition in der Zahl; ain: Zahl von dem Sinus gebildet werden soll; aout: Ergebnis.

```
API c_boolean    FC     _COS            (c_dword         alength,
                                         c_dword         adotpos,
                                         c_pointer       ain,
                                         c_pointer       aout)
```

Liefert Kosinus einer Zahl mit folgender Formel :

$$\cos x = \sum_{i=0}^{\infty} \frac{x^{2i}}{(2i)!} * (-1)^i$$

Alength: Länge der Zahl; adotpos: Kommaposition in der Zahl; ain: Zahl von dem Kosinus gebildet werden soll; aout: Ergebnis.

```
API c_boolean    FC     _TAN            (c_dword         alength,
                                         c_dword         adotpos,
                                         c_pointer       ain,
                                         c_pointer       aout,
                                         r_boolean       adiv0)
```

Liefert Tangens einer Zahl mit folgender Berechnung :

$$\tan x = \frac{\sin x}{\cos x}$$

Alength: Länge der Zahl; adotpos: Kommaposition in der Zahl; ain: Zahl von dem Tangens gebildet werden soll; aout: Ergebnis; div0: Division durch Null. Tangens von (-)Pi/2 und vielfachen nicht definiert.

```
API c_boolean    FC     _LN             (c_dword         alength,
                                         c_dword         adot,
                                         c_pointer       ain,
                                         c_pointer       aout,
                                         r_boolean       adiv0)
```

Liefert den natürlichen Logarithmus einer Zahl mit Hilfe folgender Formeln:

$$\ln x = \sum_{i=1}^{\infty} \frac{(x-1)^i}{i} * (-1)^{i+1} \qquad \text{für } 0 < x <= 2$$

Logarithmusregeln im Quellcode mit erwähnt. Alength: Länge der Zahl; adot: Kommaposition in der Zahl; ain: Zahl von dem Logarithmus gebildet werden soll; aout: Ergebnis; div0: Division durch Null.

```
boolean         FC    _PI             (c_dword        alength,
                                       c_dword        adotpos,
                                       c_pointer      aout)
```

Berechnet Pi für die vom Benutzer angegebene Genauigkeit nach folgender Formel:

$$\pi = \sum_{i=0}^{\infty} \frac{1}{16^i}\left(\frac{4}{8i+1} - \frac{2}{8i+4} - \frac{1}{8i+5} - \frac{1}{8i+6}\right)$$

Alength: Länge der Zahl; adotpos: Kommaposition in der Zahl; aout: Pi.

```
API c_boolean   FC    _ARCSIN         (c_dword        alength,
                                       c_dword        adotpos,
                                       c_pointer      pi_2,
                                       c_pointer      ain,
                                       c_pointer      aout)
```

Berechnet Arcussinus einer Zahl nach folgenden Formeln:

$$\arcsin x = x + \frac{1}{2*3}x^3 + \frac{1*3}{2*4*5}x^5 + \frac{1*3*5}{2*4*6*7}x^7 + \ldots \qquad \text{für } x^2 < 1$$

arcsin 0 = 0 ; arcsin 1 = $\frac{\pi}{2}$; arcsin -1 = -$\frac{\pi}{2}$

ArcSin x für x>1 ist nicht definiert. Alength: Länge der Zahl; adotpos: Kommaposition in der Zahl; ain: Zahl von dem Arcussinus gebildet werden soll; aout: Ergebnis; pi_2: Zahl Pi/2.

```
API c_boolean   FC    _ARCCOS         (c_dword        alength,
                                       c_dword        adotpos,
                                       c_pointer      pi,
                                       c_pointer      pi_2,
                                       c_pointer      ain,
                                       c_pointer      aout)
```

Berechnet Arcuskosinus einer Zahl nach folgenden Formeln:

$$\arccos x = \frac{\pi}{2} - \left(|x| + \frac{1}{2*3}|x|^3 + \frac{1*3}{2*4*5}|x|^5 + \frac{1*3*5}{2*4*6*7}|x|^7 + \ldots\right) \qquad \text{für } 0 < x < 1$$

$$\arccos x = \frac{\pi}{2} + \left(|x| + \frac{1}{2*3}|x|^3 + \frac{1*3}{2*4*5}|x|^5 + \frac{1*3*5}{2*4*6*7}|x|^7 + \ldots\right) \qquad \text{für } -1 < x < 0$$

arccos 0 = $\frac{\pi}{2}$; arccos 1 =0; arccos -1 = π

ArcCos x für x>1 ist nicht definiert. Alength: Länge der Zahl; adotpos: Kommaposition in der Zahl; ain: Zahl von dem Arcuskosinus gebildet werden soll; aout: Ergebnis; pi: Pi; pi_2: Pi/2.

```
API c_boolean   FC    _ARCTAN         (c_dword        alength,
                                       c_dword        adotpos,
                                       c_pointer      pi_2,
                                       c_pointer      pi_4,
                                       c_pointer      ain,
                                       c_pointer      aout)
```

Berechnet den Arcustangens einer Zahl nach folgenden Formeln:

$$\arctan x = \frac{x^1}{1} - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \frac{x^9}{9} - \frac{x^{11}}{11} + \ldots \qquad \text{für } x^2 < 1$$

$$\arctan x = \frac{\pi}{2} - \frac{1}{x^1} + \frac{1}{3x^3} - \frac{1}{5x^5} + \frac{1}{7x^7} - \ldots \qquad \text{für } x > 1$$

$$\arctan x = -\left(\frac{\pi}{2} - \frac{1}{|x|^1} + \frac{1}{3|x|^3} - \frac{1}{5|x|^5} + \ldots\right) \qquad \text{für } x < -1$$

arctan 0 = 0 ; arctan 1 = $\frac{\pi}{4}$; arctan -1 = -$\frac{\pi}{4}$

Alength: Länge der Zahl; adotpos: Kommaposition in der Zahl; ain: Zahl von dem Arcustangens gebildet werden soll; aout: Ergebnis; pi_2: Pi/2; pi_4: Pi/4.

```
API c_boolean   FC    _WURZEL         (c_dword        alength,
                                       c_dword        adotpos,
                                       c_pointer      azahl,
                                       c_pointer      awurzel,
                                       c_pointer      aout,
                                       r_boolean      adiv0)
```

Berechnet die Wurzel einer Zahl nach folgender Umformung:

$$a^{\frac{1}{x}} = y$$

$$\ln a^{\frac{1}{x}} = \ln y$$

$$\frac{1}{x} * \ln a = \ln y$$

$$e^{\frac{1}{x}*\ln a} = e^{\ln y}$$

$$e^{\frac{1}{x}*\ln a} = y$$

Die Wurzelregeln sind am Quellcode deutlich gemacht. Alength: Länge der Zahl; adotpos: Kommaposition in der Zahl; azahl: Zahl aus der die Wurzel berechnet werden soll; awurzel: die wievielte Wurzel; aout: Ergebnis; div0: Division durch 0.

```
API c_boolean   FC    _POTENZ         (c_dword        alength,
                                       c_dword        adotpos,
                                       c_pointer      azahl,
                                       c_pointer      aexponent,
                                       c_pointer      aout,
                                       r_boolean      adiv0)
```

Berechnet die Potenz einer Zahl nach folgender Umformung:

$$a^{x} = y$$

$$\ln a^{x} = \ln y$$

$$x * \ln a = \ln y$$

$$e^{x*\ln a} = e^{\ln y}$$

$$e^{x*\ln a} = y$$

Die Potenzregeln sind am Quellcode deutlich gemacht. Alength: Länge der Zahl; adotpos: Kommaposition in der Zahl; azahl: Zahl die potenziert werden soll; aexponent: Erponent; aout: Ergebnis; div0: Division durch 0.

1.1.1.4.2 Header CReal.h

CReal.h ist Wrapperklasse um einfachere Verwendung der Mathefunktionen aus der RealFunctions.cpp zu ermöglichen.
Er wurde in Teilbeleg 2 ausreichend abgehandelt.

1.1.1.5 Modul XML-Parser

Zuständige: René Kreiner, Roland Fischer

Das Modul XML-Parser dient der Umwandlung eines Strings (mit XML-Daten) in eine Baumstruktur zum gezielten Zugriff auf die einzelnen Knoten und ihrer Attribute. Ebenso wird das Aufbauen eines XML-Baumes unterstützt und das Generieren eines Strings aus diesem.

1.1.1.5.1 Header CXMLBase.h

CXMLBase ist die grundlegende Klasse des XMLParsers. Hier werden die Prototypen für das prinzipielle Verhalten gelegt.

```
#ifndef          CXMLBASE_USED
#define          CXMLBASE_USED

#include         "..\BASICS.h"


CLASSAPI                                 CXMLBase;

DEFINE_TYPES(CXMLBase)

CLASSAPI                                 CXMLBase : public CBase

  {
  public:

//Konstruktor, athreadsecured hat in dieser Anwendung keine Bedeutung.
                           CXMLBase              (c_boolean athreadsecured = false);
//Destruktor
  VIR                      ~CXMLBase             (void);
// FromString erstellt ein XML-Objekt (abgeleitet von CXMLBase) aus einem String (astring). Dabei werden die
//zu dem Objekt gehörenden Daten aus dem String gelöscht.
  VIR c_hresult      SC    FromString            (cp_CString astring);
// ToString veranlasst ein XML-Objekt, sich in einen String (astring) zu schreiben.
  VIR c_hresult      SC    ToString              (cp_CString astring);
//Clear veranlasst ein XML-Objekt, seine gesamten Daten inclusive seiner Unterobjekte zu löschen.
  VIR c_hresult      SC    Clear                 (void);
//Type liefert den genauen Typ eines XML-Objekts als ganzzahligen Code zurück.
  VIR c_dword        SC    Type                  (void);
//Get_Text liefert den Text eines XML-Objekts in einem String (astring) zurück.
  VIR c_hresult      SC    Get_Text              (cp_CString astring);
// Set_Text setzt den Text eines XML-Objekts auf den Wert eines Strings (astring).
  VIR c_hresult      SC    Set_Text              (cp_CString astring);

  protected:
//Multifunktionsstring.
  p_CString    m_string;

  public:
// Offset eines XML-Objekts in seinen übergeordneten Knoten (in Zeichen).
  dword    m_offset;

  };

//Ganzzahliger Code für die Funktion Type.
#define     XML_BASE                          0
#define     XML_DOCUMENT                      1
#define     XML_NODE                          2
#define     XML_PROCESSING_INSTRUCTION        4
#define     XML_COMMENT                       8

#endif
```

1.1.1.5.3 Header XMLFunction.h

XMLFunction enthält die Funktion xml_type. Diese ist Teil der rekursiven XML-Objekterkennung. Sie testet einen String auf die verschiedenen implementierten XML-Objektarten indem sie die FromString Routinen der ensprechenden Klassen aufruft.
Der zweite Teil der Rekursion findet in den jeweiligen Routinen statt, sofern die entsprechenden Klassen Unterknoten unterstützen.
Siehe FromString von der CXMLNode Klasse.

```
#ifndef         XMLFUNCTION_USED
#define         XMLFUNCTION_USED

#include        "..\BASICS.h"
#include        "CXML.h"
#include        "CXMLBase.h"
#include        "CXMLNode.h"
#include        "CXMLPI.h"

//Funktionalität siehe einleitende Erklärung
API c_hresult SC xml_type(p_CString astr, rp_CXMLBase abase);


#endif
```

1.1.1.5.3 Header CXMLPI.h

CXMLNode ist die grundlegende Klasse für XML-Prozessanweisungsobjekte.

```
#ifndef         CXMLPI_USED
#define         CXMLPI_USED

#include        "..\BASICS.h"


CLASSAPI                            CXMLPI;
DEFINE_TYPES(CXMLPI)

CLASSAPI                            CXMLPI : public CXMLBase
  {
  public:

//Konstruktor, athreadsecured hat in dieser Anwendung keine Bedeutung.
                        CXMLPI              (c_boolean athreadsecured = false);
//Destruktor
  VIR                   ~CXMLPI             (void);

// FromString erstellt ein XML-Objekt (abgeleitet von CXMLBase) aus einem String (astring). Dabei werden die
//zu dem Objekt gehörenden Daten aus dem String gelöscht.
  VIR c_hresult     SC    FromString          (cp_CString astring);
// ToString veranlasst ein XML-Objekt, sich in einen String (astring) zu schreiben.
  VIR c_hresult     SC    ToString            (cp_CString astring);

//Type liefert den genauen Typ eines XML-Objekts als ganzzahligen Code zurück.
  VIR c_dword       SC    Type                (void);

  };

#endif
```

1.1.1.5.4 Header CXMLNode.h

CXMLNode ist die grundlegende Klasse für normale XML-Knotenobjekte.

```
#ifndef          CXMLNODE_USED
#define          CXMLNODE_USED

#include         "..\BASICS.h"

CLASSAPI                                CXMLNode;
DEFINE_TYPES(CXMLNode)

CLASSAPI                                CXMLNode : public CXMLBase
  {
  public:

//Konstruktor, athreadsecured hat in dieser Anwendung keine Bedeutung.
                          CXMLNode              (c_boolean athreadsecured = false);
//Destruktor
  VIR                     ~CXMLNode             (void);

// FromString erstellt ein XML-Objekt (abgeleitet von CXMLBase) aus einem String (astring). Dabei werden die
//zu dem Objekt gehörenden Daten aus dem String gelöscht.
  VIR c_hresult      SC   FromString            (cp_CString astring);
// ToString veranlasst ein XML-Objekt, sich in einen String (astring) zu schreiben.
  VIR c_hresult      SC   ToString              (cp_CString astring);
//Clear veranlasst ein XML-Objekt, seine gesamten Daten inklusive seiner Unterobjekte zu löschen.
  VIR c_hresult      SC   Clear                 (void);

//Type liefert den genauen Typ eines XML-Objekts als ganzzahligen Code zurück.
  VIR c_dword        SC   Type                  (void);

//Get_Name liefert den Name eines XML-Objekts in einem String (astring) zurück.
  VIR c_hresult      SC   Get_Name              (cp_CString astring);
// Set_Name setzt den Name eines XML-Objekts auf den Wert eines Strings (astring).
  VIR c_hresult      SC   Set_Name              (cp_CString astring);

 //Get_Att_Val liefert das Attribut eines XML-Objekts in einem String (astring) zurück.
  VIR c_hresult      SC   Get_Att_Val           (cp_CString aattr, cp_CString aval);
// Set_Att_Val setzt das Attribut eines XML-Objekts auf den Wert eines Strings (astring).
  VIR c_hresult      SC   Set_Att_Val           (cp_CString aattr,
                                                 cp_CString aval);
//Get_Sub_Count liefert die Anzahl der Unterobjekte eines  XML-Objekts zurück.
  VIR c_dword        SC   Get_Sub_Count         (void);
//Get_Sub liefert den Pointer des Unterobjekts mit dem Index aindex zurück
 VIR c_hresult       SC   Get_Sub               (c_dword    aindex, rp_CXMLBase axml);
//Add_New_Sub_Node erstellt einen neuen Knoten und gibt ihn in axml zurück
  VIR c_hresult      SC   Add_New_Sub_Node      (rp_CXMLNode axml);

  private:

//Name eines XML-Objektes
  p_CString   m_name;
//Unterobjekjte eines XML-Objektes
  p_CObjectList m_subnodes;
//Attribute und Attributwerte eines XML-Objektes
  p_CDoubleStringList m_attributes;

  };

#endif
```

1.1.1.5.5 Header CXML.h

CXML repräsentiert ein komplettes XML-Dokument. Es ist der Einstiegspunkt für das Modul Formelparser.

```
#ifndef         CXML_USED
#define         CXML_USED

#include        "..\BASICS.h"
#include        "CXMLBase.h"
#include        "CXMLPI.h"


CLASSAPI                                CXML;
DEFINE_TYPES(CXML)

CLASSAPI                                CXML : public CXMLBase
  {

  public:
//Konstruktor, athreadsecured hat in dieser Anwendung keine Bedeutung.
                        CXML                    (c_boolean athreadsecured = false);
//Destruktor
 VIR                    ~CXML                   (void);
// FromString erstellt ein XML-Objekt (abgeleitet von CXMLBase) aus einem String (astring). Dabei werden die
//zu dem Objekt gehörenden Daten aus dem String gelöscht.
  VIR c_hresult     SC    FromString            (cp_CString astring);
// ToString veranlasst ein XML-Objekt, sich in einen String (astring) zu schreiben.
  VIR c_hresult     SC    ToString              (cp_CString astring);

 //Type liefert den genauen Typ eines XML-Objekts als ganzzahligen Code zurück.
  VIR c_dword       SC    Type                  (void);
//Get_Sub_Count liefert die Anzahl der Unterobjekte eines  XML-Objekts zurück.
  VIR c_dword       SC    Get_Sub_Count         (void);
//Get_Sub liefert den Pointer des Unterobjekts mit dem Index aindex zurück
  VIR c_hresult      SC    Get_Sub                  (c_dword     aindex, rp_CXMLBase
axml);
//Add_New_Sub_Node erstellt einen neuen Knoten und gibt ihn in axml zurück
  VIR c_hresult     SC    Add_New_Sub_Node      (rp_CXMLNode axml);
//Add_New_Sub_Node erstellt einen neuen Knoten und gibt ihn in axml zurück
  VIR c_hresult     SC    Add_New_PI            (rp_CXMLPI axml);

  private:
//gesamten Knoten-Objekte des Baumes
  p_CObjectList m_items;

  };
#endif
```

1.1.2 MathClient

Die Module des MathClients exportieren keine Routinen. Sie werden nur intern benutzt.

1.1.2.4 Modul MathStuff

Zuständiger: Thomas Weise

Aufgrund einer schweren Erkrankung von Rico Roßberg musste der Teamleiter einspringen, um die Arbeit am MathClient fortzusetzen. Da er jedoch mit der Planung und der Struktur des Kodes des Rico Roßberg sowie der Programmiersprache Visual Basic nicht familiär war, konnte nur eine eine provisorische Lösung zur Umwandlung von der Infixschreibweise mathematischer Ausdrücke zu gültigem MathML-Text entwickelt werden. Diese weicht von der Planung in Teilbeleg 2 stark ab. Der betreffende Teil des Beleg wurde von Herrn Roßberg erstellt. Der Teamleiter war nicht näher mit den exakten inneren Überlegungen vertraut, weshalb er sich außerstande sah, sie 1:1 zu realisieren.

Das Modul dient der Umwandlung der üblichen Infixschreibweise mathematischer Ausdrücke in gültiges MathML.

Zu verwenden ist die Routine math2xml, welche einen solchen Infixausdruck erhält und das MathML-Pendant ausgibt, sowie eine Liste (IDS) mit allen Unterausdrücken des Infixausdrucks füllt, damit diese dann im Teilergebnisfenster angezeigt werden können.

```
Function math2xml(ByVal S As String, ByRef IDS() As String) As String
```

1.2 Kommentierte Modulquelltexte

1.2.1 MathServer

1.2.1.1 Modul Basics

1.2.1.1.1 Kode CBasics.cpp

```
#include        "..\BASIC_Classes.h"


//Konstruktor, athreadsecured erzeugt Sperrmutex, wird in dieser Anwendung nicht benötigt
CBase::CBase                (c_boolean athreadsecured)
  {
  m_refcount    = 1;
  m_state       = 0;
  if(athreadsecured)
    {
    m_mutex     = CreateMutexW(0, false, 0);
    }
  else m_mutex = INVALID_HANDLE_VALUE;
  }



//Destruktor
CBase::~CBase               (void)
  {
  if(m_mutex != INVALID_HANDLE_VALUE)
    {
    CloseHandle(m_mutex);
    m_mutex    = INVALID_HANDLE_VALUE;
    }
  }

//gibt den aktuellen Wert der Statusvariable zurück
c_dword         SC  CBase::State(void)
  {
  return(m_state);
  }



//prüft die Statusvariable auf eine bestimmte Bitkombination
c_boolean       SC  CBase::GetState(c_dword astate)
  {
  return((m_state & astate) == astate);
  }
```

```
//setzt/löscht eine Bitkombination als Status, verhindert, dass der DEATH_STATE
//überschrieben werden kann

#pragma warning (push)
#pragma warning (disable : 4035)
c_dword       SC  CBase::SetState(c_dword astate, c_boolean aset)
  {
  __asm
    {
    and   astate,   ~FORBIDDEN_STATE
    mov      ecx,   this
    mov      eax,   [ecx+4]
    mov      edx,   aset
    test     edx,   edx
    jz       del
    or       eax,   astate
    jmp       ok
del:
    not   astate
    and      eax,   astate
ok:
    mov  [ecx+4],   eax
    }
  }
#pragma warning (pop)


//gibt den aktuellen Wert des Referenzzählers wieder
c_dword       SC   CBase::RefCount             (void)
  {
  return m_refcount;
  }


//erhöht den Referenzzähler um eins
//diese Methode ist absolut threadsicher, sie gibt einen Wert > 0 wieder,
//wenn das Objekt noch existiert
//und 0, wenn es sich bereits im Zustand der Destruktion befindet
//Die Threadsicherheit wird erzeugt durch die Variable m_State (an this+04h), siehe release
#pragma warning (push)
#pragma warning (disable : 4035)
c_dword       SC  CBase::AddRef(void)
  {
  _asm
    {
            mov                 eax,       1
            mov                 ecx,       this
    lock    xadd            [ecx+8],       eax
            inc                 eax
            test            [ecx+4],       DEATH_STATE
            jz                 ende
            xor                 eax,       eax
    ende:
    }
  }
#pragma warning (pop)
```

```
//Release verringert den Referenzzähler um eins, und gibt das Objekt frei, wenn 0 erreicht wird
//Diese Methode ist neben AddRef einer der Dreh- und Angelpunkte unserer Objekthierarchie, und deshalb auch
//threadsicher ausgelegt.

c_dword       SC  CBase::Release(void)
  {
  dword   retval   = 0;
  dword   destruct = 0;
  _asm
    {
//Zuerst testen wir, ob der Referenzzähler bereits <= 0 ist.
//In diesem Fall wird der Destruktor mit Sicherheit bereits (evtl. in einem anderen
//Thread oder Fiber bereits ausgeführt, und wir verlassen die Routine sofort.
            mov               ecx,       this
            mov               eax,       [ecx+8]
            cmp               eax,       0
            jg                ok
            xor               eax,       eax
            jmp               res
//Indem wir 1 auf eax schaffen, und mit xadd addieren (xadd tauscht vor Addition, in eax steht dann der alte
//Wert der des Referenzzählers) können wir durch nachfolgendes Dekrementieren von eax feststellen, ob wir
//den Destruktor auslösen müssen.
//Aus der Gleichlaufsicht betrachtet, kann folgendes geschehen: ein Release führt die xadd Routine, die zum
//Destruktor führt, genau eine Instruktion vor oder nach unserem  xadd aus, dann erhalten entweder wir oder er
//-1, aber nur einer 0
//Wird genau eine Instruktion davor das xadd des AddRef ausgeführt, so wird kein Destruktor ausgeführt und
//AddRef gibt 2 zurück, was stimmt. (Das Objekt wird ja nicht freigegeben).
//Erfolgt AddRef's xadd genau eine Instruktion danach, so erhält AddRef beim xadd 0 und wird
//entsprechend reagieren. Das Objekt wird dann nämlich bereits freigegeben, neue Referenzen wären
//inkonsistent.
            ok:
            mov               eax,       -1
    lock    xadd            [ecx+8],     eax
            dec               eax

            res:
            mov             retval,      eax

            test              eax,       eax
            jnz              ende
//An dieser Stelle ist klar, dass das Objekt freigegeben werden muss - es muss nur noch abgeklärt werden,
//ob wir uns bereits innerhalb des Destruktors befinden. Es ist nämlich durchaus möglich, dass es in einem
//Destruktor nötig ist, das Objekt noch einmal als Parameter an eine Funktion zu übergeben. Diese könnte
//AddRef und Release auslösen. Dieses würde zu einem Destruktorlauf führen, da diese im Release innerhalb
//der Funktion wieder aufgerufen werden würde, und die Funktion und somit das Release und somit sich selbst
//und so weiter und so fort aufrufen würde.
//Hier kommt der DEATH_STATE ins Spiel. Wir sichern uns gegen andere Threads und Fibers ab, indem wir
//xchg verwenden, und den State erstmal generel auf DEATH_STATE setzen.
//Dann kopieren wir den alten Zustand wieder dazu.
//Jeder parallele Handlungsstrang hat dadurch entweder vor uns DETH_STATE gesetzt, oder erhält unseren
//DEATH_STATE als State-Variable
//Der Destruktor wird dann nur aufgerufen, wenn DEATH_STATE noch nicht gesetzt war.
            mov               eax,       DEATH_STATE
     lock   xchg              eax,       [ecx+4]
            or              [ecx+4],     eax
            test              eax,       DEATH_STATE
            jnz              ende

            mov           destruct,      0xffffffff

            ende:
    }
  if(destruct)
    {
    delete this;
    }
  return(retval);
  }
```

```
//Wie gesagt, Enter und Release waren für eine Cooperation zwischen verschiedenen Rechnern bzw.
//Anwendungen am selben Problem gedacht und dienen prinzipiell dazu, dass ein Objekt threadsensitiven Kode
//wegsperren kann. Bei der aktuellen Projektkonfiguration spielen sie keine Rolle.

void          SC    CBase::Enter                (void)
  {
  AddRef();
  if(m_mutex != INVALID_HANDLE_VALUE) wait(m_mutex);
  }

void          SC    CBase::Leave                (void)
  {
  if(m_mutex != INVALID_HANDLE_VALUE) ReleaseMutex(m_mutex);
  Release();
  }


//Mit den Operatoren new und delete binden wir unsere Wrapper für die Windows API ein
c_pointer       CBase::operator new      (unsigned int asize)
  {
  pointer x = allocate(asize);
  if(x==0) throw;
  else return x;
  }


void            CBase::operator delete   (void* amemory)
  {
  free(amemory);
  }
```

1.2.1.2 Kode CSocket.cpp

```
#include        "..\BASIC_Classes.h"


wsadata   _wsadata;
dword      wsalock = 0;

//Diese Routine sollte bei Programmstart aufgerufen werden, sie initialisiert die Windows Netzwerkumgebung
c_hresult     FC   StartWSA(void)
  {
  hresult retval = S_OK;

  dword l = InterlockedIncrement(wsalock);
  if(l == 1)
    {
    zero(&_wsadata, sizeof(_wsadata));
    if(WSAStartup(MAKEWORD(2, 2), _wsadata) == 0)
      {
      }
    else {
         retval  = wsaerror();
         wsalock = 0;
         }
    }

  return retval;
  }


//Diese Routine sollte bei Programmende aufgerufen werden, sie deinitialisiert die Windows Netzwerkumgebung
c_hresult     FC   CloseWSA(void)
  {
  hresult retval = S_OK;

  dword l = InterlockedDecrement(wsalock);
  if(l == 0)
    {
    if(WSACleanup()!=0) retval = wsaerror();
    zero(&_wsadata, sizeof(wsadata));
    }

  return retval;
  }


//Im Konstruktor werden die Variablen initialisiert.
CSocket::CSocket          (c_boolean athreadsecured) : CStream(athreadsecured)
  {
  m_event  = WSA_INVALID_EVENT;
  m_socket = INVALID_SOCKET;
  zero(&m_address, sizeof(m_address));
  }


//...und im Destruktor deinitialisiert
CSocket::~CSocket         (void)
  {
  m_socket = INVALID_SOCKET;
  zero(&m_address, sizeof(m_address));
  }

//Gibt wieder, ob das Socket geöffnet ist.
c_boolean     SC   CSocket::IsOpen              (void)
  {
  if(m_socket != INVALID_HANDLE_VALUE) return true;
  else return false;
  }
```

```
//Gibt wieder, ob vom Socket gelesen werden kann.
c_boolean      SC    CSocket::CanRead                (void)
  {
  if(IsOpen() && GetState(CSOCKET_CLIENT)) return true;
  else return false;
  }

//Gibt wieder, ob auf das Socket geschrieben werden kann.
c_boolean      SC    CSocket::CanWrite               (void)
  {
  if(IsOpen() && GetState(CSOCKET_CLIENT)) return true;
  else return false;
  }

//Schließt das Socket und das dazugehörige Ereignis.
c_hresult      SC    CSocket::Close                  (void)
  {
  hresult retval = S_OK;

  if(m_socket != INVALID_SOCKET)
    {
    if(closesocket(m_socket) != 0) retval = wsaerror();
    m_socket = INVALID_SOCKET;
    }

  if(m_event != WSA_INVALID_EVENT)
    {
    if(!(WSACloseEvent(m_event))) retval = worst(retval, wsaerror());
    m_event = WSA_INVALID_EVENT;
    }

  zero(&m_address, sizeof(m_address));
  SetState(CSOCKET_ALL, false);

  return retval;
  }

//Setzt das Socket als Server. Dazu wird zuerst die Netzwerkadresse aufgelöst, an die gebunden werden soll
//dann wird ein Event für eingehende Verbindungen erstellt (zu dem wir später noch kommen) und schließlich
//an die Adresse gebunden und gelauscht.

c_hresult      SC    CSocket::Server                 (p_CString abindaddress)
  {
  hresult retval = Close();

  if(succeeded(retval))
    {
    m_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if(m_socket != INVALID_SOCKET)
      {
      if(abindaddress) abindaddress->ToIPAddress(m_address);
      else m_address.in_addr.dwords[0] = INADDR_ANY;
      m_address.family = AF_INET;
      m_event = WSACreateEvent();

      if(m_event != WSA_INVALID_EVENT)
        {
        if(WSAEventSelect(m_socket, m_event, FD_ACCEPT) == 0)
          {
          if(bind(m_socket, &m_address, sizeof(m_address)) == 0)
            {
            if(listen(m_socket, SOMAXCONN) == 0)
              {
              SetState(CSOCKET_SERVER, true);
              goto end;
              }
            }
          }
```

```
        }
      }
    retval = wsaerror();
    }

end:
  if(failed(retval)) Close();

  return retval;
  }


//Wartet auf eine eingehende Verbindung und akzeptiert sie. Als Abbruchbedingung kann die Callback-Routine
//aterminateproc angegeben werden, die aller 15ms mit aparam aufgerufen wird. Gibt sie true zurück, so wird
//das Warten auf eingehende Verbindungen abgebrochen.

c_hresult     SC   CSocket::Accept                (rp_CSocket anewsock,
                                                   c_pointer  aterminateproc,
                                                   c_dword    aparam)
  {
  hresult     retval   = S_OK;
  dword       waitres;
              anewsock = 0;

  if(m_socket != INVALID_SOCKET)
    {
    if(GetState(CSOCKET_SERVER))
      {
      if(aterminateproc)
        {
//so lange Abbruchbedingung nicht erfüllt
        while ( ! (CAST(boolean_method, aterminateproc)(aparam)) )
          {
//maximal 15ms warten auf eingetroffene Verbindung
          while( (waitres = WSAWaitForMultipleEvents(1, &m_event, true, 15, true))
                                              == WAIT_IO_COMPLETION) ;
//wenn neue Verbindung, gehe zu ok, ansonsten erst mal einen anderen Thread ranlassen
          if(waitres == WSA_WAIT_EVENT_0) goto ok;
          SwitchToThread();
          }
        retval = S_FALSE;
        goto end;
        }
      else
        {
//ist terminateproc nicht angegeben, so kann ewiglich gewartet werden
        while ((waitres = WSAWaitForMultipleEvents(1, &m_event, true, WSA_INFINITE,
                          true)) == WAIT_IO_COMPLETION) ;
        }
ok:   if(waitres == WSA_WAIT_EVENT_0)
        {
//neues Warten ermöglichen
        if(WSAResetEvent(m_event))
          {
          if(retval == S_OK)
            {
//neues Socket erstellen
            CSocket* x = new CSocket();
            if(x)
              {
              dword l = sizeof(x->m_address);
//Verbindung akzeptieren
              x->m_socket = accept(m_socket, x->m_address, l);

              if(x->m_socket != INVALID_SOCKET)
                {
//Die Event-Konfiguration dieses Sockets wurde vom Serversocket (mit m_event) übernommen, und muss
//rückgesetzt werden
                if(WSAEventSelect(x->m_socket, 0, 0) == 0)
```

```
                    {

                    dword z = 0;
//Socket in den Blocking-Mode schalten
                    if(ioctlsocket(x->m_socket, FIONBIO, &z) == 0)
                      {
                      x->SetState(CSOCKET_CLIENT, true);
                      anewsock = x;
                      goto end;
                      }
                    }
                  }
                retval = wsaerror();
                x->Release();
                }
              else retval = E_OUTOFMEMORY;
             }
           }
         else retval = wsaerror();
         }
       else retval = wsaerror();
       }
     else retval = E_FAIL;
     }
   else retval = E_FAIL;
end:
   return retval;
   }
```

```
//Baut eine Klienten-Verbindung zu der Adresse aconnectaddress auf

C_hresult     SC   CSocket::Client              (p_CString aconnectaddress)
   {
   hresult retval = Close();

   if(succeeded(retval))
     {

//Socket-Handle erzeugen
     m_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

     if(m_socket != INVALID_SOCKET)
       {

//Netzwerkadresse auflösen
       if(aconnectaddress) aconnectaddress->ToIPAddress(m_address);
       else m_address.in_addr.dwords[0] = LOCAL_ADDRESS;

       m_address.family = AF_INET;

//Verbindung aufbauen
       if(connect(m_socket, &m_address, sizeof(m_address)) == 0)
         {
         SetState(CSOCKET_CLIENT, true);
         }
       else retval = wsaerror();
       }
     else retval = wsaerror();
     }

   if(failed(retval)) Close();

   return retval;
   }
```

```
//asize Bytes von adata auf das Socket schreiben, geschrieben wurden awritten, geht nur bei Klienten

c_hresult     SC   CSocket::Write                (c_pointer adata, c_dword asize,
                                                 r_dword awritten)
  {
  hresult retval = CStream::Write(adata, asize, awritten);

  if(retval == E_NOTIMPL)
    {
    retval   = S_OK;
    dword  s = asize;
    dword  l;
    p_byte x = (p_byte)adata;
//Es ist möglich, dass nicht alle Daten auf einmal versendet werden können, warum auch immer.
//Deshalb wird ein einer Schleife so lange gesendet, bis entweder das Ende des Sendebuffers erreicht wurde
//oder ein Fehler auftrat.
      do
        {
        l = send(m_socket, (c_pointer)x, s, 0);
        if(l == SOCKET_ERROR) retval = wsaerror();
        else
          {
          s        -= l;
          x        += l;
          awritten += l;
          }
        }
      while ((succeeded(retval)) & (s > 0) & (l > 0));
    }
  return retval;
  }
//vom Socket asize Bytes in adata lesen, aread = Anzahl gelesener Bytes

c_hresult     SC   CSocket::Read                 (c_pointer adata, c_dword asize,
                                                  r_dword   aread)
  {
  hresult retval = CStream::Read(adata, asize, aread);
  if(retval == E_NOTIMPL)
    {
    retval   = S_OK;
    aread = recv(m_socket, adata, asize, 0);
    }
  return retval;
  }


//Die Adresse des Sockets in einen String schreiben
c_hresult     SC   CSocket::Address              (p_CString aaddress)
  {
  hresult retval = S_OK;
  if(aaddress)
    {
    if(IsOpen())
      {
      retval = aaddress->FromIPAddress(m_address);
      }
    else
      {
      aaddress->Clear();
      retval = E_FAIL;
      }
    }
  else retval = E_INVALIDARG;
  return retval;
  }
```

1.2.1.3 Kode CThread.cpp

```
#include        "..\BASIC_Classes.h"

//Hier ist die einzige Stelle im Code, wo tatsächlich der Sperrparameter athreadsecured sowie Enter
//und Leave ins Spiel kommen

//Konstruktor, erzeugt die nötigen Events and Handles
CThread::CThread                (p_CThread aowner) : CObjectList(true)
  {
  m_owner = 0;

//Wenn es einen übergeordneten Thread gibt, dann füge dich in die Liste seiner Subthreads ein
  if(aowner)
    {
    PURE_SET(m_owner, aowner)
    if(m_owner)
      {
      aowner->Add(this);
      }
    }

//Jeder Thread in Windows folgt der Aufrufkonvention _stdcall. Er erhält stets einen 32-Bit-Parameter.
//Jede Methode einer Klasse in VC++ erhält als ersten Parameter einen 32-Bit-Wert, this, den Zeiger
//auf das Objekt.
//Diesen Mechanismus nutzen wir, um eine virtuelle, überschreibbare Klassenmethode in als
//Thread-Routine nutzen zu können. Dazu müssen wir ihren Index in der VTBL, der virtuellen Methodentabelle
//des Objekts (von CThread abgeleitet, daher immer selber Platz) wissen (COBJECTLIST_VTBL_LEN).
//Jede Klasse besitzt in ihren Daten an Offset 0 einen unsichtbaren Parameter, den VTBL-Pointer.
//Er zeigt auf eine Liste von Methodenadresse. Wir dereferenzieren also zweimal, und erhalten somit die
//Einsprungadresse des Threads. Als Parameter übergeben wir this. So wird beim ersten Resume die Routine
//MainThread aufgerufen. Diese dient als Framework für Execute, was dann beliebig überschrieben werden
//kann.
  m_handle              = CreateThread(0, 0, VTBL_ITEM(COBJECTLIST_VTBL_LEN),
                                       this, CREATE_SUSPENDED, m_id);

  m_subthreadsevent     = CreateEventW(0, true, true, 0);

  m_exitcode            = S_FALSE;

  SetState(CTHREAD_SUSPENDED | CTHREAD_FIRST, true);
  }


CThread::~CThread               (void)
  {
  Terminate();
  WaitFor();

  CloseHandle(m_handle);
  m_handle          = 0;
  CloseHandle(m_subthreadsevent);
  m_subthreadsevent = 0;

  if(m_owner) m_owner->Remove(this);
  SET_NULL(m_owner)
  }
```

```
c_hresult     SC   CThread::MainThread          (void)
  {
  hresult retval;
//Im Mainthread sichern wir zuersteinmal mit AddRef die Instanz, setzen dann das Running-Flag und führen
//die Benutzerdefinierte Execute-Funktion aus.
//Wenn diese beendet wurde, so entfernen wir den Thread aus der Liste der laufenden Threads des Owners.
//(übergeordneter Thread).
  AddRef();
  SetState(CTHREAD_RUNNING, true);
  retval     = Execute();
  m_exitcode = retval;
  if(m_owner)
    {
    retval   = worst(retval, m_owner->Remove(this));
    }
  SET_NULL(m_owner)

//Anschließend löschen wir das Running-Flag und heben die Instanzsicherung auf.
//Durch das Instanzsichern wird es möglich, die Referenz auf den Thread direkt bei seiner Ausführung zu
//löschen, er wird sich zu gegebener Zeit selbst freigeben.
  SetState(CTHREAD_RUNNING, false);
  Release();
  return retval;
  }


//Resume startet den Thread
c_hresult     SC   CThread::Resume              (void)
  {
  hresult retval = E_UNEXPECTED;

  if( (m_id == 0) || (GetCurrentThreadId() != m_id) )
    {
    if(m_handle)
      {
      if(GetState(CTHREAD_TERMINATED | CTHREAD_FIRST))
        {
        if(CloseHandle(m_handle))
          {
          m_handle = 0;
          }
        else retval = error();
        }
      else
        {
        SetState(CTHREAD_FIRST, false);
        dword i = ResumeThread(m_handle);
        if(i == MAX_DWORD)
          {
          retval = error();
          }
        else
          {
          if(retval <= 1) SetState(CTHREAD_SUSPENDED, false);
          retval = S_OK;
          }
        }
      }
    else retval = S_FALSE;
    }

  return retval;
  }
```

```
//Suspend stellt ihn auf "Pause"
c_hresult      SC    CThread::Suspend              (void)
  {
  hresult retval = E_UNEXPECTED;

  if( (m_id == 0) || (GetCurrentThreadId() != m_id)  && (m_handle))
    {
    if(SuspendThread(m_handle) == MAX_DWORD)
      {
      retval = error();
      }
    else
      {
      SetState(CTHREAD_SUSPENDED, true);
      retval = S_OK;
      }
    }
  return retval;
  }

c_hresult      SC    CThread::Execute              (void)
  {
//Execute wird von abgeleiteten Klassen überschrieben, um den sinnvollen Code zu implementieren
  return S_OK;
  }

//Therminate setzt das Terminate-Flag für Execute und terminiert auch alle Sub-Threads
c_hresult      SC    CThread::Terminate            (void)
  {
  hresult retval = S_OK;
  SetState(CTHREAD_TERMINATED, true);
  if(GetState(CTHREAD_FIRST) && (m_handle))
    {
    if(CloseHandle(m_handle))
      {
      m_handle = 0;
      }
    else retval = error();
    }
  retval = worst(retval, TerminateSubThreads());
  return retval;
  }

//TherminateSubThreads terminiert alle Sub-Threads
c_hresult      SC    CThread::TerminateSubThreads  (void)
  {
  hresult retval = S_OK;
  Enter();
  dword l = Length();
  if(l)
    {
    pp_CThread d = (pp_CThread)DataPtr();
    if(d)
      {
      for(dword i = 0; i<l; i++)
        {
        if(d[0])
          {
          retval = worst(retval, d[0]->Terminate());
          }
        d++;
        }
      }
    }
  Leave();
  return retval;
  }
```

```
//Wartet auf alle Sub-Threads

c_hresult     SC   CThread::WaitForSubThreads     (c_dword atime)
  {
  return wait(m_subthreadsevent, atime);
  }

//Wartet auf Hauptthread
c_hresult     SC   CThread::WaitForMainThread     (c_dword atime)
  {
  if(GetState(CTHREAD_FIRST)) return S_OK;

  if( (m_id == 0) || (GetCurrentThreadId() != m_id) )
    {
    return wait(m_handle, atime);
    }

  return E_UNEXPECTED;
  }


//Wartet eine festgelegte Zeit auf den Haupt- und seine Sub-Threads

c_hresult     SC   CThread::WaitFor              (c_dword atime)
  {
  if(GetState(CTHREAD_FIRST)) return S_OK;

  if( (m_id == 0) || (GetCurrentThreadId() != m_id) )
    {
    dword   handles[2] = {m_handle, m_subthreadsevent};

    dword   i;

    while ( (i=WaitForMultipleObjectsEx(2, (p_dword)&handles, true, atime, true))
                                == WAIT_IO_COMPLETION) ;

    switch(i)
      {
      case MAX_DWORD      : return error();
      case WAIT_TIMEOUT   : return S_FALSE;
      default             : return S_OK;
      }

    }

  return E_UNEXPECTED;
  }


//Beendet den Thread durch das Terminate-Signal und wartet auf sein tatsächliches (sicheres) Ende

c_hresult     SC   CThread::TerminateAndWait     (void)
  {
  hresult retval = E_UNEXPECTED;

  if( (m_id == 0) || (GetCurrentThreadId() != m_id) )
    {
    retval = Terminate();
    if(succeeded(retval))
      {
      retval = WaitFor();
      }
    }

  return retval;
  }
```

```
c_hresult      SC    CThread::ExitCode              (void)
  {
  return m_exitcode;
  }


c_dword        SC    CThread::Id                    (void)
  {
  if((m_handle) && (m_id)) return m_id;
  return 0;
  }

//untergeordnete Bedeutung haben
c_hresult      SC    CThread::SetPriority           (c_dword apriority)
c_dword        SC    CThread::GetPriority           (void)
c_boolean      SC    CThread::Started               (void)
c_boolean      SC    CThread::Running               (void)
c_boolean      SC    CThread::Suspended             (void)
c_boolean      SC    CThread::Terminated            (void)
void           SC    CThread::SwitchToOther         (void)


//Mit den folgenden Methoden wird gewährleistet, dass neue Sub-Threads stets in einer Liste des Hauptthreads
//gesichert werden. Dadurch ist es möglich, dass der Hauptthread auf sie wartet, wenn er das Terminate-Signal
//erhält.
//Nur so kann z.B. später im Server bei dessen Beendigung auch die korrekte Abfertigung aller gerade laufender
//Aufträge garantiert werden. (Neue werden dann natürlich nicht mehr angenommen.)
c_hresult      SC    CThread::SetLength             (c_dword alength)
  {
  hresult retval = CObjectList::SetLength(alength);

  if(succeeded(retval))
    {
    boolean x;
    if(alength > 0) x = ResetEvent(m_subthreadsevent);
    else            x =   SetEvent(m_subthreadsevent);
    if(!(x)) retval = error();
    }

  return retval;
  }


c_hresult      SC    CThread::Delete                (c_dword aindex)
  {
  hresult retval;

  Enter();
  retval = CObjectList::Delete(aindex);
  Leave();

  return retval;
  }


c_hresult      SC    CThread::Remove                (c_pointer aitem)
  {
  hresult retval;

  Enter();
  retval = CObjectList::Remove(aitem);
  Leave();

  return retval;
  }
```

1.2.1.2 Modul Engine

1.2.1.2.1 Kode Engine_Processor.cpp

```
#include        "..\Engine.h"


CProcessor::CProcessor           (p_CStream aindata,
                                  p_CStream aoutdata,
                                  p_CThread aowner) : CThread(aowner)
  {
  m_indata  = new CHTTPStream(aindata); //wir erstellen HTTPStreams um die
  m_outdata = new CHTTPStream(aoutdata); //Eingangsstreams herum
  }


CProcessor::~CProcessor          (void)
  {
  SET_NULL(m_indata) //und geben sie hier wieder frei
  SET_NULL(m_outdata)
  }

c_hresult     SC   CProcessor::Execute              (void)
  {
  dword retval = S_OK;

  if((m_indata) && (m_outdata))
    {
    p_CString ais = new CString();
    if(ais)
      {
      dword encoding;

      retval = m_indata->ReadStr(ais, encoding);

      if(succeeded(retval))
        {
        p_CString aos = new CString();

        if(aos)
          {
          p_CString x = new CString();
          p_CString y = new CString();

          m_indata->GetHeader(x);

          dword s, e;
          dword fractals = 100;
          dword decimals = 100;
          qword z;

          s = x->FindUnicode(_S("decimals")); //herausfinden, wieviele Dezimal-
          if(s != NOT_FOUND)                  //stellen gewünscht
            {
            s = x->FindUnicode(_S("\""), s);
            if(s != NOT_FOUND)
              {
              e = x->QuoteEnd(s);
              if(succeeded(x->ExtractNum(z, s+1, e-1))) decimals = (dword)z;
              }
            }

          s = x->FindUnicode(_S("fractals")); //und wieviele Nachkommastellen
          if(s != NOT_FOUND)
            {
```

```
            s = x->FindUnicode(_S("\""), s);
            if(s != NOT_FOUND)
              {
              e = x->QuoteEnd(s);
              if(succeeded(x->ExtractNum(z, s+1, e-1))) fractals = (dword)z;
              }
            }


          x->Clear();

//Ausgabedaten warden mit Hilfe des Formelparsers berechnet
          retval = Formula_Parse(ais, aos, decimals, fractals);
//Aus Gründen der Systemsicherheit und Transparenz wird eine umfassende Fehler-
//behandlung durchgeführt
          if(succeeded(retval))
            {
            x->AddAscii("200 OK");
            }
          else
            {
            x->AddAscii("800 ERROR");
            }

          m_outdata->SetServerHeader(x, y);

          retval = m_outdata->WriteStr(aos, encoding);

          SET_NULL(x)
          SET_NULL(y)
          SET_NULL(aos);
          }
        else retval = E_OUTOFMEMORY;
        }

      SET_NULL(ais);
      }
    }
  else retval = E_INVALIDARG;

  return retval;
  }
```

1.2.1.2.2 Kode Engine_Server.cpp

```
#include    "..\Engine.h"


CServer::CServer              (p_CString aport) : CThread(0)
  {
//wir erstellen ein neues Serversocket und lassen es an aPort lauschen,
//wie uns geheißen
  m_socket = new CSocket();
  if(m_socket) m_socket->Server(aport);
  }


CServer::~CServer             (void)
  {
//und geben es hier wieder frei
  SET_NULL(m_socket)
  }

c_hresult     SC   CServer::Execute              (void)
  {
  hresult retval = S_OK;

  if(m_socket)
    {
    p_CSocket    data = 0;
    p_CProcessor p    = 0;
//als Terminated-Callback geben wir dem Socket unsere Terminated-Routine mit.
//beendet der Nutzer das Programm, so erhält der MainThread ein WM_QUIT
//dann wird unser Thread hier mit Terminate() zur Beendigung aufgefordert
//dadurch werden keine Verbindungen mehr angenommen
//die bestehenden (einzelne Processors) aber noch abgearbeitet
    while ( (retval = m_socket->Accept(data, VTBL_ITEM(CTHREAD_TERMINATED_INDEX),
(dword)this)) == S_OK)
      {

      if(!(Terminated()) && (data))
        {
        p = 0;
//für jede Eingangsverbindung wird ein Processor gestartet
        p = new CProcessor(data, data, CAST(p_CThread, this));
        p->Resume();
        SET_NULL(p)
        }

      SET_NULL(data)
      }

    }

  return retval;
  }
```

1.2.1.3 Modul Formelparser

1.2.1.3.1 Kode Formel.cpp

```
#include        "Formel.h"

//Die eigentliche Arbeitsfunktion des Formelparsers
//Der übergebene Knoten node muß ein "Apply"-Knoten sein
//Die Ergebnisse werden _unsortiert_ zum String aresult zusammengefügt
    c_hresult   FC  Recursive_Parsing   (p_CXMLNode node, p_CString aresult,
                                         rp_CReal aretval, p_CConstants aconsts,
                                         c_dword anumlength, c_dword adotpos)
      {
      hresult retval = S_OK;      //Rückgabewert der Funktion

      aretval = 0;

      if((node) && (aresult))
        {
        CString* a= new CString();      //Pufferstring
        CString* ID= new CString();     //Enthält die ID des Knotens

        retval=a->AddUnicode(_S("ID"));
        retval=node->Get_Att_Val(a,ID); //Wert des ID-Attributes ermitteln

        dword CC=node->Get_Sub_Count(); //Anzahl der Söhne des Knotens

        CXMLNode* op = 0;  //Erster Sohn (Operator)

        retval = node->Get_Sub(0, CAST(rp_CXMLBase, op));

        if(succeeded(retval))
          {
          op->Get_Name(a); //Operatorname ermitteln

          CReal* ops[2];   //Feld für Operanden
          ops[0] = 0;
          ops[1] = 0;
          CXMLNode* s = 0;//Sohnknoten für Operanden
          CString* b = new CString();   //Pufferstring

          //Operanden suchen (1 oder 2 Stück), Typ und Werte ermitteln (Zwischenergebnisse)
  for(dword i = 1; (i < CC) && (i < 3) && (i < node->Get_Sub_Count()); i++)
            {
            if(succeeded(node->Get_Sub(i, CAST(rp_CXMLBase, s))))
              {
              s->Get_Name(b);      //Typ des Operanden

                      //untergeordneter Klammerausdruck ("Apply")
              if( (b->CompareUnicode(_S("Apply")) & CMP_MATCH) != 0)
                {
                          //Selbstaufruf für untergeordneten Ausdruck
    retval = Recursive_Parsing(s, aresult, ops[i-1], aconsts, anumlength, adotpos);
                }
              else
                {          //Zahl
                if( (b->CompareUnicode(_S("cn")) & CMP_MATCH) != 0)
                  {
                  s->Get_Text(b);       //Zahl ermitteln
                  ops[i-1] = FromString(anumlength, adotpos, b);
                  }
                else
                  { //Konstante (nur noch Pi...)
                  s->Get_Text(b);
                  if( (b->CompareUnicode(_S("PI")) & CMP_MATCH) != 0 )
```

```
                {
                ops[i-1] = PI(aconsts);
        //Kein weiteres ELSE, da der MathClient sonst nichts weiter erzeugt...
                                //...und wir von dessen Fehlerfreiheit ausgehen...
                }
              }
            }
          }
        }

          //Select Case (bzw. dessen Äquivalent in dieser furchtbaren PS hier) für den Operator
          //Ungültige Zwischenergebnisse (Fehler) untergeordneter Teilausdrücke brauchen nicht
          //beachtet werden, da die mathematischen Funktionen die Fehlermeldung durchreichen

          //Betrag
    if((a->CompareUnicode(_S("abs")) & CMP_MATCH) != 0)
      {
      aretval = Abs(ops[0]);
      } else
                //Arkuskosinus
    if((a->CompareUnicode(_S("arccos")) & CMP_MATCH) != 0)
      {
      aretval = ArcCos(ops[0], aconsts);
      } else
                //Arkussinus
    if((a->CompareUnicode(_S("arcsin")) & CMP_MATCH) != 0)
      {
      aretval = ArcSin(ops[0], aconsts);
      } else
                //Arkustangens
   if((a->CompareUnicode(_S("arctan")) & CMP_MATCH) != 0)
      {
      aretval = ArcTan(ops[0], aconsts);
      } else
                //Kosinus
  if((a->CompareUnicode(_S("cos")) & CMP_MATCH) != 0)
      {
      aretval = Cos(ops[0], aconsts);
      } else
                //Division
           if((a->CompareUnicode(_S("divide")) & CMP_MATCH) != 0)
      {
      aretval = Div(ops[0], ops[1]);
      } else
                //e^
  if((a->CompareUnicode(_S("exp")) & CMP_MATCH) != 0)
      {
      aretval = E_X(ops[0]);
      } else
                //Fakultät
  if((a->CompareUnicode(_S("factorial")) & CMP_MATCH) != 0)
      {
      aretval = Factorial(ops[0]);
      } else
                //Natürlicher Logarithmus
  if((a->CompareUnicode(_S("ln")) & CMP_MATCH) != 0)
      {
      aretval = Ln(ops[0]);
      } else
                //Subtraktion
           if((a->CompareUnicode(_S("minus")) & CMP_MATCH) != 0)
      {
      aretval = Sub(ops[0], ops[1]);
      } else
                //Addition
    if((a->CompareUnicode(_S("plus")) & CMP_MATCH) != 0)
      {
      aretval = Add(ops[0], ops[1]);
      } else
```

```
                //Potenz
    if((a->CompareUnicode(_S("power")) & CMP_MATCH) != 0)
      {
      aretval = Potenz(ops[0], ops[1]);
      } else
    //quotient
    if((a->CompareUnicode(_S("quotient")) & CMP_MATCH) != 0)
      {
      aretval = IDiv(ops[0], ops[1]);
      } else
                //Wurzel
    if((a->CompareUnicode(_S("root")) & CMP_MATCH) != 0)
      {
      aretval = Wurzel(ops[1], ops[0]);
      } else
                //Sinus
    if((a->CompareUnicode(_S("sin")) & CMP_MATCH) != 0)
      {
      aretval = Sin(ops[0], aconsts);
      } else
                //Tangens
               if((a->CompareUnicode(_S("tan")) & CMP_MATCH) != 0)
      {
      aretval = Tan(ops[0], aconsts);
      } else
                //Multiplikation
    if((a->CompareUnicode(_S("times")) & CMP_MATCH) != 0)
      {
      aretval = Mul(ops[0], ops[1]);
      } else
                //Multiplikation
    if((a->CompareUnicode(_S("log")) & CMP_MATCH) != 0)
      {
      aretval = Log(ops[0], ops[1]);
      }
         //Fehler
         else
      {
      aretval = new CReal(anumlength, adotpos);
      aretval->Invalidate();
      }

    ToString(aretval, b);   //Zwischenergebnis in String umwandeln
    if(succeeded(retval))
                //Ergebnisstring zusammensetzen
      {
      aresult->AddUnicode(_S("<cn id=\""));
      aresult->AddStr(ID);
      aresult->AddUnicode(_S("\">"));
      aresult->AddStr(b);
      aresult->AddUnicode(_S("</cn>"));
      }

    SET_NULL(s)       //Pointer auflösen
    SET_NULL(ops[0])
    SET_NULL(ops[1])
    SET_NULL(b)
    }

  SET_NULL(op)

  SET_NULL(a);
  SET_NULL(ID);
  }
```

1.2.1.4 Modul Math

1.2.1.4.1 Kode RealFunctions.cpp

```
API c_pointer   FC    _NEW          (c_dword        alength)
                                                  //erstellt Pointer auf Zahl (nimmt Specher)
  {
  pointer retval = 0;
  if(alength)
    {
    retval = allocate(4 * alength);                                  //Speicher holen
    }
  return retval;
  }

API c_pointer   FC    _COPY         (c_dword        alength,
                                     c_pointer      aoriginal)
                                                  //kopiert Pointer zu Pointer
  {
  pointer retval = _NEW(alength);
  if(retval)
    {
    if(retval) move(retval, aoriginal, alength * 4);                 //kopieren
    }
  return retval;
  }

API void        FC    _COPYBITS     (c_dword        alength,
                                     c_pointer      aoriginal,
                                     c_dword        aleft,
                                     c_dword        aright,
                                     c_pointer      aoutdata)
                                                  //kopiert bitweise Pointer zu Pointer
  {
  if((aleft>=aright) && (aleft<alength*32) && (aright>=0))       //Start und Ende prüfen
    {
    dword min = (aright / 32);                                   //Startblock ermitteln
    dword mov = ((aleft-aright)/32)+1;                           //Anzahl der Blöcke ermitteln
    zero(aoutdata, min*4);                                       //Löschen der außerhalb der Blöcke
    zero(&(((p_dword)aoutdata)[min+mov]), (alength-min-mov)*4);  // liegenden Bits
    move(&(((p_dword)aoutdata)[min]), &(((p_dword)aoriginal)[min]), mov*4); //Bitcopy
    ((p_dword)aoutdata)[min] &= (0xffffffff << (aright % 32) );  //restliche Bits
    ((p_dword)aoutdata)[min+mov-1] &= (0xffffffff >> (32-((aleft+1)%32)));  //löschen
    }
  }

API void        FC    _FREE         (r_pointer      anum)
                                                  //gibt Pointer (Speicher) frei
  {
  free(anum);
  }
```

```
API void        FC    _NOT          (c_dword        length,
                                     c_pointer      aindata,
                                     c_pointer      aoutdata)
                                                          //Bildet 1er-Kompl.
  {
  __asm
    {
    test      ecx,ecx                                     //Länge der Zahl=0?
    jz        ende
    mov       esi,aindata
    mov       edi,aoutdata
    cld
  looper:
    lodsd                                   //32-Bit Block nach eax lesen, LesePos+=32
    not       eax                                       //32-Bit Block negieren
    stosd                                   //schreiben auf aoutdata, SchreibePos+=32
    loop      looper
  ende:
    }
  }

API void        FC   _NOT2          (c_dword        alength,
                                     c_pointer      adata,
                                     c_pointer      aoutdata)
  {
  __asm
    {
    test      ecx,ecx
    jz        ende
    mov       esi,edx
    mov       edi,aoutdata
    cld
looper:
    lodsd       //suchen bis erstes Mal eine 1 in einem 32Bit-Block kommt, dann Sprung zu found
    test      eax,eax
    jnz       found
    stosd                                           //sonst Nullblöcke speichern
    loop      looper
    jmp       ende
found:
    neg       eax                                           //2er-Kompl. bilden
    stosd                                                   //speichern
    dec       ecx
    jz        ende
looper2:
    lodsd                               //alle weiteren Blöcke einfach negieren und speichern
    not       eax
    stosd
    loop      looper2
ende:
    }
  }

API c_boolean   FC   _POSITIVE      (c_dword        alength,
                                     c_pointer      aindata)
                                                          //liefert true wenn MSB=0
  {
  return (((((p_dword)aindata)[alength-1]) & DWORD_MSB) == 0);
  }

API c_boolean   FC   _NEGATIVE      (c_dword        alength,
                                     c_pointer      aindata)
                                                          //liefert true wenn MSB=1
  {
  return (((((p_dword)aindata)[alength-1]) & DWORD_MSB) != 0);
  }
```

```
API c_boolean   FC   _TEST_BIT       (c_dword        alength,
                                     c_dword        abit,
                                     c_pointer      adata)
                                                        //gibt zurück ob Bit gesetzt
  {
  return (((((p_dword)adata)[abit / 32]) & (1 << (abit % 32))) != 0);
  }

API void        FC   _INS_BIT        (c_dword        alength,
                                     c_dword        abitnum,
                                     c_boolean      aooz,
                                     c_pointer      adata)
                                              //setzt oder löscht Bit an angebener Position
  {
  if(aooz) (((p_dword)adata)[abitnum / 32]) |= (1 << (abitnum % 32));       //setzen
  else (((p_dword)adata)[abitnum / 32]) &= ~(1 << (abitnum % 32));          //löschen
  }

API c_dword     FC   _MSB            (c_dword        alength,
                                     c_pointer      adata)
  {
  __asm
    {
    mov       ebx,0xffffffff                                          //ebx = -1
    test      ecx,ecx                                          //wenn Länge der Zahl=0
    jz        ende                                                   //sprung zu ende
    mov       esi,edx                                       //esi auf LSB zeigen lassen..
    mov       edx,ecx                        //Offset bestimmen: 32-Bit-Blöckeanzahl nach edx
                                                        //(Addressierung in 32Bit-Blöcken)
    dec       edx                                                  //edx um eins verringern
    shl       edx,2                                  //edx mal vier (Addressierung in Bytes)
    add       esi,edx                                                 //LSB+Offset=MSB
    std                                                          // für esi-1 bei lodsd
looper:                                        //Suche des 32Bit-Blockes welcher das MSB trägt
    lodsd                                      //32Bit-Block in eax laden (erstes Mal MSB-Block)
    test      eax,eax                            //Setzt das zeroflag wenn die Oderverknüpfung
                                                          //der Bits von bitweisen(eax&&eax)=0
    jnz       found       //=>sobald die erste '1' in einem 32 Bit-Block erscheint sprung zu found
    loop      looper                                  //sonst.. weiter bei looper, ecx-1
    jmp       ende2     //wenn Schleife bis zum Ende läuft (LSB erreicht) liefert die Funktion false
found:                                          //Suche nach der 1 innerhalb des 32Bit-Blockes
    dec       ecx                    //Offset ermitteln: ecx auf MSB des 32Bit-Block zeigen lassen
                                                        //(Addressierung in 32Bit-Blöcken)
                                       //(welcher das MSB der Zahl hat) (Addressierung in Bytes)
    mov       ebx,ecx                                                 //ebx ergibt sich aus ecx
    shl       ebx,5                          //ebx*32 = MSB Position der Zahl (Addressierung in Bits)
    bsr       ecx,eax                              //sucht innerhalb des 32Bit-Blockes nach dem
                                  //MSB (fängt bei Bit 31 an) und speichert Index in ecx (Bitaddressierung)
                                                        //wird kein Bit gefunden ist Zeroflag 1
    jz        ende2                                                  //..und Sprung zu ende2
    add       ebx,ecx                                              //..sonst Bitindex ermitteln
ende2:
    cld                                                                  //wegen std
ende:
    mov       eax,ebx                                //Bitindex wird auf Rückgabewert eax geschafft
    }
  }
```

```
API c_dword     FC    _LSB            (c_dword        alength,
                                       c_pointer      adata)
                                                  //funktioniert nach dem Prinzip von _MSB
  {
  __asm
    {
    mov       ebx,0xffffffff
    test      ecx,ecx
    jz        ende
    mov       esi,edx
    mov       edx,ecx
    cld
looper:
    lodsd       //Testen von rechts nach links, der 32-Bit-Blöcke, wenn gefunden weiter mit found:
    test      eax,eax
    jnz       found
    loop      looper
    jmp       ende
found:                                  //Suche innerhalb des 32Bit-Blockes und ermitteln der Position
    sub       edx,ecx
    mov       ebx,edx
    shl       ebx,5
    bsf       ecx,eax
    jz        ende
    add       ebx,ecx
ende:
    mov       eax,ebx
    }
  }

API c_dword     FC    _CMP            (c_dword        alength,
                                       c_pointer      aindata1,
                                       c_pointer      aindata2)
  {
  __asm
    {
    mov       ebx,CMP_SAME                                      //bei gleichen Pointern
    cmp       edx,aindata2
    jz        ende
    mov       ebx,CMP_EQUAL                                     //wenn Länge der Zahlen Null
    test      ecx,ecx
    jz        ende
    mov       esi,aindata1
    mov       edi,aindata2
    mov       edx,ecx
    dec       edx
    shl       edx,2
    add       esi,edx
    add       edi,edx
    std
    lodsd         //obersten 32Bit-Block vorzeichenbehaftet prüfen, wenn gleich weiter bei looper
    mov       edx,dword ptr [edi]
    dec       ecx
    cmp       eax,edx
    jl        smaller
    jg        bigger
    test      ecx,ecx
    jz        ende2
    sub       edi,4
    std
looper:                                                       //weitere Blöcke vorzeichenlos prüfen
    lodsd
    mov       edx,dword ptr [edi]
    cmp       eax,edx
    jb        smaller
    ja        bigger
    sub       edi,4
    loop      looper
    jmp       ende2
```

```
smaller:                                                    //zuweisen der Rückgabedaten
    mov       ebx,CMP_SMALLER
    jmp       ende2
bigger:
    mov       ebx,CMP_BIGGER
ende2:
    cld
ende:
    mov       eax,ebx
    }
  }

API c_boolean   FC    _IZ           (c_dword        alength,
                                     c_pointer       aindata)
  {
  __asm
    {
    test      ecx,ecx                                                    //alength=0?
    jz        ende                                                     //ja... sprung ende
    cld                                                          //directionflag:=0 (incdec=2)
    mov       edi,edx                                                  //edi:=edx
    shl       ecx,1                                                      //ecx*2
    xor       eax,eax                                                   //eax:=0
    rep       scasw                                          //rep: wiederholt solang zf<>0 und
                                                          //cx>0; scasw: vergleich eax mit [edi]
    jnz       ende                                                    //nein->ende
    not       eax                                                       //true
ende:
    }
  }

API c_boolean   FC    _SHL_1        (c_dword        alength,
                                     c_pointer       aindata,
                                     c_pointer       aoutdata)
  {
  __asm
    {
        xor       eax,eax                                          //eax auf 0 setzen
        test      ecx,ecx                                                //alength=0?
        jz        ende                                           //ja... sprung ende
        mov       esi,aindata                                      //esi:= aindata
        mov       edi,aoutdata                                    //edi:= aoutdata
        cld                                                 //directionflag:=0 (incdec=4)
        clc                                                          //c-flag:=0
looper:
        lodsd                                                  //eax:= [esi]; esi+4
        rcl       eax,1              //temp:=msb, linksschieben, lsb:=cf, cf:=temp
        stosd                                                  //[edi]:= eax; edi+4
        loop      looper    //# 4byte-blöcke (ecx) um 1 verringern, sprung zu looper
        jb        carry                                         //spring wenn cf=1
        xor       eax,eax                                                //eax=0
        jmp       ende                                         //sprung ende
carry:
        mov       eax,0xffffffff                                      //eax=true
ende:
    }
  }
```

```
__forceinline c_boolean   FC    _SHL_N_MOD32     (c_dword         alength,
                                                  c_dword         adigits,
                                                  c_pointer       aindata,
                                                  c_pointer       aoutdata)
                                                        //Schieben um n Bits (n<32)
  {
  byte        rdigits;
  byte        ldigits;
  dword       mask;
  dword       counter;
  __asm
    {
    xor       eax,eax
    test      ecx,ecx
    jz        ende                                  //Testen auf Zahl mit Länge 0
    pushf                                                       //Flags sichern
    mov       counter,ecx                     //Anzahl der 32-Bit-Blöcke sichern
    mov       esi,aindata
    mov       edi,aoutdata
    mov       cl,dl
    mov       mask,0xffffffff
    shr       mask,cl
    not       mask
    mov       rdigits,32                     //Linksverschiebung der Sicherung
    sub       rdigits,dl                                        //berechnen
    mov       ldigits,dl                   //Rechtsverschiebung sichern
    xor       ebx,ebx                                             //ebx:=0
    cld                                        //directionflag:=1 (dec=4)
looper:
    lodsd                                                 //eax:= [esi]; esi-4
    mov       cl,rdigits
    mov       edx,eax
    and       edx,mask                                  //n_Rausschiebebits
    shr       edx,cl                                              //sichern
    mov       cl,ldigits
    shl       eax,cl                                          //Rechtsschieben
    or        eax,ebx                                 //a_Rausschiebebits in eax
einbringen
    mov       ebx,edx                       //a_Rausschiebebits:=n_Rausschiebebits
    stosd                                                 //[edi]:= eax; edi-4
    dec       counter
    jnz       looper
    test      ebx,ebx
    jnz       carry                                     //spring wenn cf=1
    xor       eax,eax                                                 //eax=0
    jmp       ende2                                         //sprung ende
carry:
    mov       eax,0xffffffff                                        //eax=true
ende2:
    popf                                              //Flags wiederherstellen
ende:
    }
  }
```

```
__forceinline  c_boolean   FC   _SHL_N_32        (c_dword         alength,
                                                  c_dword         adigits,
                                                  c_pointer       aindata,
                                                  c_pointer       aoutdata)
  {
  boolean retval;
  _asm
    {                                                    //Überprüfen ob Überlauf entsteht
    mov       retval,0
    test      ecx,ecx                                                        //alength=0?
    jz        ende                                                    //ja... sprung ende
    cmp       edx,ecx
    jae       ende
    mov       ebx,ecx
    mov       eax,ecx
    sub       eax,edx
    shl       eax,2
    cld
    mov       edi,aindata
    add       edi,eax
    mov       ecx,edx
    shl       ecx,1
    xor       eax,eax
    rep       scasw                                  //rep: wiederholt solang zf<>0 und
                                                    //cx>0; scasw: vergleich eax mit [edi]
    jz        ende
    not       retval
ende:
    }                                                                  //Schieben der Blöcke
  move( &((p_dword)(aoutdata))[adigits], aindata, (alength-adigits)*4);
  zero(aoutdata, adigits*4);
  return retval;
  }

API c_boolean   FC    _SHL_N          (c_dword         alength,
                                       c_dword         adigits,
                                       c_pointer       aindata,
                                       c_pointer       aoutdata)
  {
  boolean retval = false;
  dword d        = adigits / 32;
  if(d) retval = _SHL_N_32(alength, d, aindata, aoutdata);          //Blöcke schieben
  else move(aoutdata, aindata, alength*4);
                                                                      //Bits schieben
  retval = _SHL_N_MOD32(alength, adigits % 32, aoutdata, aoutdata) || retval;
  return retval;
  }

API c_boolean   FC    _SHR_1          (c_dword         alength,
                                       c_pointer       aindata,
                                       c_pointer       aoutdata)
                                            //Schiebt um 1 bit rechts
  {
  __asm
    {
    xor       eax,eax                                          //eax auf 0 setzen
    test      ecx,ecx                                                  //alength=0?
    jz        ende                                              //ja... sprung ende
    pushf                                                        //Flags sichern
    mov       esi,aindata                                        //esi:= aindata
    mov       edi,aoutdata                                      //edi:= aoutdata
    mov       edx,ecx                                                 //ebx:=ecx
    dec       edx                                                       //ebx-1
    shl       edx,2                                                     //ebx*4
    add       esi,edx                                                 //esi+ebx
    add       edi,edx                                                 //edi+ebx
    std                                          //directionflag:=1 (dec=4)
    clc                                                            //c-flag:=0
```

```
looper:
    lodsd                                                   //eax:= [esi]; esi-4
    rcr       eax,1                   //temp:=lsb, rechtsschieben, msb:=cf, cf:=temp
    stosd                                                   //[edi]:= eax; edi-4
    loop      looper        //# 4byte-blöcke (ecx) um 1 verringern, sprung zu looper
    jb        carry                                         //spring wenn cf=1
    xor       eax,eax                                                 //eax=0
    jmp       ende2                                       //sprung ende
carry:
    mov       eax,0xffffffff                                  //eax=true
ende2:
    popf
ende:                                                   //Flags wiederherstellen
    }
  }

__forceinline c_boolean   FC    _SHR_N_MOD32     (c_dword         alength,
                                                  c_dword         adigits,
                                                  c_pointer       aindata,
                                                  c_pointer       aoutdata)
                                                //Rechtsschieben von n Bits (n<32)
  {
  byte        rdigits;
  byte        ldigits;
  dword       mask;
  dword       counter;
  __asm
    {
    xor       eax,eax
    test      ecx,ecx
    jz        ende                                  //Testen auf Zahl mit Länge 0
    pushf                                                   //Flags sichern
    mov       counter,ecx                         //Anzahl der 32-Bit-Blöcke sichern
    mov       ebx,ecx
    dec       ebx
    shl       ebx,2
    mov       esi,aindata
    mov       edi,aoutdata
    add       esi,ebx                             //esi aufs MSDW der aindata setzen
    add       edi,ebx
    mov       cl,dl
    mov       mask,1
    shl       mask,cl                                       //Sicherungsmaske
    dec       mask                                                  //erstellen
    mov       ldigits,32                    //Linksverschiebung der Sicherung
    sub       ldigits,dl                                            //berechnen
    mov       rdigits,dl                          //Rechtsverschiebung sichern
    xor       ebx,ebx                                               //ebx:=0
    std                                             //directionflag:=1 (dec=4)
looper:
    lodsd                                                   //eax:= [esi]; esi-4
    mov       cl,ldigits
    mov       edx,eax
    and       edx,mask                                    //n_Rausschiebebits
    shl       edx,cl                                                //sichern
    mov       cl,rdigits
    shr       eax,cl                                              //Rechtsschieben
    or        eax,ebx                                   //a_Rausschiebebits in eax
einbringen
    mov       ebx,edx                           //a_Rausschiebebits:=n_Rausschiebebits
    stosd                                                   //[edi]:= eax; edi-4
    dec       counter
    jnz       looper
    test      ebx,ebx
    jnz       carry                                         //spring wenn cf=1
    xor       eax,eax                                                 //eax=0
    jmp       ende2                                       //sprung ende
```

```
carry:
    mov       eax,0xffffffff                                    //eax=true
ende2:
    popf                                                  //Flags wiederherstellen
ende:
    }
  }

__forceinline c_boolean   FC    _SHR_N_32       (c_dword         alength,
                                                 c_dword         adigits,
                                                 c_pointer       aindata,
                                                 c_pointer       aoutdata)
                                                //Rechtsschieben von 32Bit-Blöcken
  {
  boolean retval;
  _asm
    {                                                      //Prüfen auf Unterlauf
    mov       retval, 0
    test      ecx,ecx                                           //alength=0?
    jz        ende                                        //ja... sprung ende
    cmp       edx,ecx
    jae       ende
    cld                                               //directionflag:=0 (incdec=2)
    mov       edi,aindata
    mov       ebx,ecx
    mov       ecx,edx
    shl       ecx,1                                                //ecx*2
    xor       eax,eax                                             //eax:=0
    rep       scasw                                  //rep: wiederholt solang zf<>0 und
                                                  //cx>0; scasw: vergleich eax mit [edi]
    jz        ende                                              //nein->ende
    not       retval
ende:
    }
                                                        //Rechtsschieben der Blöcke
  move(aoutdata, &((p_dword)(aindata))[adigits], (alength-adigits)*4);
  zero(&((p_dword)(aoutdata))[alength-adigits], adigits*4);
  return retval;
  }

API c_boolean   FC    _SHR_N          (c_dword         alength,
                                       c_dword         adigits,
                                       c_pointer       aindata,
                                       c_pointer       aoutdata)
  {
  dword d = adigits / 32;
  boolean retval = false;
  if(d) retval = _SHR_N_32(alength, d, aindata, aoutdata);          //Blöckeschieben
  else move(aoutdata, aindata, alength*4);
                                                                    //Bits schieben
  retval = _SHR_N_MOD32(alength, adigits % 32, aoutdata, aoutdata) || retval;
  return retval;
  }

API c_boolean   FC    _ADD            (c_dword         alength,
                                       c_pointer       aindata1,
                                       c_pointer       aindata2,
                                       c_pointer       aoutdata)
  {
  __asm
    {
        xor       eax,eax                                     //eax auf 0 setzen
        test      ecx,ecx                                           //alength=0?
        jz        ende                                      //ja... sprung ende
        mov       esi,aindata1                                 //esi:= aindata1
        mov       ebx,aindata2                                 //ebx:= aindata2
        mov       edi,aoutdata                                 //edi:= aoutdata
        cld                                           //directionflag:=0 (incdec=4)
```

```
        clc                                                     //c-flag:=0
looper:
        lodsd                                                   //eax:= [esi]; esi+4
        mov     edx,[ebx]                                       //edx:= [ebx]
        adc     eax,edx                                         //eax:=eax+edx+cf
        stosd                                                   //[edi]:= eax; edi+4
        pushf                                                   //flags (cf) in ah sichern
        add     ebx,4                                           //ebx+4
        popf                                                    //flags aus ah wiederherstellen
        loop    looper          //anzahl der 4byte-blöcke (ecx) um 1 verringern
        mov     eax,0                                           //eax=0
        jno     ende
        not     eax
ende:
    }
  }

API c_boolean   FC    _SUB          (c_dword        alength,
                                    c_pointer       aindata1,
                                    c_pointer       aindata2,
                                    c_pointer       aoutdata)
  {
  __asm
    {
        xor     eax,eax                                         //eax auf 0 setzen
        test    ecx,ecx                                         //alength=0?
        jz      ende                                            //ja... sprung ende
        mov     esi,aindata1                                    //esi:= aindata1
        mov     ebx,aindata2                                    //ebx:= aindata2
        mov     edi,aoutdata                                    //edi:= aoutdata
        cld                                             //directionflag:=0  (incdec=4)
        clc                                                     //c-flag:=0
  looper:
        lodsd                                                   //eax:= [esi]; esi+4
        mov     edx,[ebx]                                       //edx:= [ebx]
        sbb     eax,edx                                         //eax:=eax-(edx+cf)
        stosd                                                   //[edi]:= eax; edi+4
        pushf                                                   //flags (cf) in ah sichern
        add     ebx,4                                           //ebx+4
        popf                                                    //flags aus ah wiederherstellen
        loop    looper    //# 4byte-blöcke (ecx) um 1 verringern, sprung zu looper
        mov     eax,0                                           //eax=0
        jno     ende
        not     eax
ende:
    }
  }

API c_boolean   FC    _MUL          (c_dword        alength,
                                    c_dword         adot,
                                    c_pointer       aindata1,
                                    c_pointer       aindata2,
                                    c_pointer       aoutdata,
                                    r_boolean       aunderflow)
  {
  boolean boolette;
  boolean kompl;
  int i;
  int msb_op2;
  int lsb_op1;
  int lsb_op2;
  dword exactlyshift;
  pointer op1;
  pointer op2;
  boolette=false;
  kompl=false;
  aunderflow=false;
```

```
op1=_COPY (alength, aindata1);
op2=_COPY (alength, aindata2);
zero(aoutdata, alength*4);
if(_NEGATIVE(alength, op1))                                    //Zahlen positiv machen
  {                                                              //feststellen welches
  _NOT2(alength, op1, op1);                                  //Vorzeichen Ergebnis bekommt
  kompl=true;
  }
if(_NEGATIVE(alength, op2))
  {
  _NOT2(alength, op2, op2);
  kompl=!(kompl);
  }
lsb_op1=_LSB(alength, op1);
lsb_op2=_LSB(alength, op2);
if(!(lsb_op1==NOT_FOUND) && !(lsb_op2==NOT_FOUND))
  {
  if(lsb_op1<lsb_op2)                  //Exactlyshift (Rechtsverschiebung) ermitteln
    {
    exactlyshift=lsb_op1;
    }
  else
    {
    exactlyshift=lsb_op2;
    }
  _SHR_N(alength, exactlyshift, op1, op1);                              //Rechtsschieben
  _SHR_N(alength, exactlyshift, op2, op2);
  lsb_op2-=exactlyshift;
  msb_op2=_MSB(alength, op2);
  i=msb_op2;
                                                          //Berechnung mittels SHIFT&ADD
  while((i>=0) && !(boolette))
    {
    if(_SHR_1(alength, op2, op2))
      {
      boolette = _ADD(alength, op1, aoutdata, aoutdata);
      }
    boolette=_SHL_1(alength, op1, op1) || boolette;
    i--;
    }
  integer s  = (2*exactlyshift) - adot;
  if(s<0)
    {
    if(s <= (-(32*(integer)alength)))          //Unterlauf prüfen und Ergebnis..
                                              //..zuweisen nach Korrekturschiebung
      {
      aunderflow = true;
      zero(aoutdata, alength * 4);
      }
    else aunderflow=_SHR_N(alength, -s, aoutdata, aoutdata);
    }
  else
    {
    if(s >= (32*(integer)alength)) boolette = true;              //Überlauf prüfen
    else boolette = _SHL_N(alength, s, aoutdata, aoutdata);  //Korrekturschiebung
    }
  if(kompl) _NOT2(alength, aoutdata, aoutdata);          //Vorzeichen einarbeiten
  }
_FREE (op1);
_FREE (op2);
return boolette;
}
```

```
API c_boolean   FC    _DIV            (c_dword         alength,
                                       c_dword         adot,
                                       c_pointer       aindata1,
                                       c_pointer       aindata2,
                                       c_pointer       aoutdata,
                                       r_boolean       aunderflow,
                                       r_boolean       div0)
                                                  //berechnet aindata1/aindata2
  {
  boolean aoverflow;
  boolean kompl;
  int     s;
  int     outpos;
  int     msb_op1;
  int     msb_op2;
  dword   exactlyshift;
  dword   o;
  pointer op1;
  pointer op2;
  pointer temp;
  aoverflow = false;
  kompl     = false;
  op1       = _COPY (alength, aindata1);                          //op1=aindata1
  op2       = _COPY (alength, aindata2);                          //op2=aindata2
  zero(aoutdata, alength*4);
          //Berträge der Operanden werden gebildet und das zu erwartende Vorzeichen ermittelt.
  if(_NEGATIVE(alength, op1))
    {
    _NOT2(alength, op1, op1);
    kompl = true;
    }
  if(_NEGATIVE(alength, op2))
    {
    _NOT2(alength, op2, op2);
    kompl = !(kompl);
    }
  msb_op1 = _MSB(alength, op1);
  msb_op2 = _MSB(alength, op2);
  div0    = (msb_op2 == NOT_FOUND);
  if(!(msb_op1 == NOT_FOUND) && (!div0))                          // x/0 abfangen
    {
    if((_CMP(alength, op1, op2) & CMP_MATCH) != 0)                // x/x abfangen
      {
      _INS_BIT (alength, adot, 1, aoutdata);
      }
    else
      {
      s            = msb_op1 - msb_op2;                           //Outputpos bestimmen
      outpos       = adot + s;
                   //Beide Operanden werden in Abhängigkeit des jeweiligen MSB um genau
                                  exactlyshift Stellen nach links geschoben um die Division möglichst
                                                                              genau durchzuführen.
      exactlyshift = (alength * 32) - 2;
      o            = exactlyshift;
      if(msb_op1 > msb_op2)  exactlyshift -= msb_op1;
      else                   exactlyshift -= msb_op2;
      if(outpos > (integer)o)
        {
        aoverflow = true;
        }
      else
        {
        if(outpos < 0) aunderflow = true;
        else
          {
          dword s1 = exactlyshift;
          dword s2 = exactlyshift;
          if(s <= 0) s1 -= s;
          else       s2 += s;
```

```
        _SHL_N(alength, s1, op1, op1);          //Exactlyshift , op2 mit seinem msb auf auf
        _SHL_N(alength, s2, op2, op2);                  // gleiche Pos wie op1 schieben

        temp  = _NEW(alength);
        do
          {                     // Op2 wird von op1 subtrahiert, wenn ergebnis pos. dann wird
                          //ergibt sich op1 aus ergebnis und in aoutdata wird eine 1 geschrieben.
          _SUB (alength, op1, op2, temp);
          if(_POSITIVE (alength, temp))
            {
            move(op1, temp, 4*alength);
            _INS_BIT (alength, outpos, true, aoutdata);
            }
                                                    // Danach Shiftright von op2 und Vorgang
          _SHR_1 (alength, op2, op2);
          outpos--;
          }
                                                  //wiederholen bis op1,op2=0 oder Unterlauf.
        while ((!_IZ(alength, op2)) && (!_IZ(alength, op1)) && (outpos >= 0));
        _FREE(temp);
        if((outpos <= 0) && (!_IZ(alength, op1))) aunderflow=true;
        }
      }
    }
                                                          //erwartetes Vorzeichen einbringen
  if((!div0) && kompl) _NOT2(alength, aoutdata, aoutdata);
  }
_FREE (op1);
_FREE (op2);
return aoverflow;
}

__forceinline c_boolean   FC    _STR_MUL_10         (c_dword         alength,
                                                     c_pointer       aindata,
                                                     c_pointer       aoutdata,
                                                     r_boolean       aunderflow)
                                                              //multipliziert Zahl mit 10
{
boolean boolette;
pointer op;
boolette=false;
op=_COPY (alength, aindata);
zero(aoutdata, alength*4);
boolette=_SHL_1(alength, op, op) || boolette;                          //10[dez]=1010[bin]
boolette=_ADD(alength, op, aoutdata, aoutdata) || boolette;
boolette=_SHL_N(alength, 2, op, op) || boolette;
boolette=_ADD(alength, op, aoutdata, aoutdata) || boolette;
_FREE(op);
return boolette;
}

__forceinline c_boolean   FC    _ADD_DWORD          (c_dword         alength,
                                                     c_pointer       aindata1,
                                                     c_dword         aindata2,
                                                     c_pointer       aoutdata)
{
__asm
  {
      xor       eax,eax
      test      ecx,ecx
      jz        ende
      mov       esi,aindata1
      mov       edi,aoutdata
      xor       edx,edx
      cld
      lodsd
      add       eax,aindata2
      stosd
      jnc       ende2
```

```
looper:
        lodsd
        adc       eax,edx                    //Stellenweises addieren mit Übertrag
        stosd
        jnc       ende2
        loop      looper
ende2:
        mov       eax,0
        jno       ende
        not       eax
ende:
    }
  }

API c_hresult  FC    _STR2NUM          (c_string      astring,
                                        c_dword       astringlength,
                                        c_character   astringdot,
                                        c_dword       anumlength,
                                        c_dword       anumdotpos,
                                        c_pointer     anumnew,
                                        r_boolean     aoverflow,
                                        r_boolean     aunderflow)
                                                  //Wandelt einen String zu einer Zahl.
                                      //Indem der String von links nach rechts ausgelesen wird.
                                        //Jede Ziffernstelle der Strings wird auf die Zahl addiert,
                                                  //Zahl *10 bis Stringende erreicht wird.
                                      //Zum Schluß Zahl / (10^(Strignachkommastellen)) und
                                                  //evtl. 2er-Kompl. Bilden.
  {
  hresult retval = S_OK;
  boolean kompl;
  boolean div0;
  aoverflow  = false;
  aunderflow = false;
  if(astring && anumnew)
    {
    if(anumlength > 0)
      {
      dword dotposinstring;
      for(dotposinstring = 0;
          (dotposinstring<astringlength) && (astring[dotposinstring] !=
          astringdot); dotposinstring++);              //Kommapos in String ermitteln
      dword i;
      i=1;
      if (astring[0]==_S('-'))                                  //Vorzeichen merken
        kompl=true;                                   //Startpkt in String festlegen
      else
        {
        kompl=false;
        if(astring[0]!=_S('+')) i=0;
        }
                                                    //Vorkommastellen berechnen
      aoverflow=_ADD_DWORD(anumlength, anumnew, astring[i]-_S('0'), anumnew);
      i++;
      for(i; i < dotposinstring; i++)
        {
        aoverflow=_STR_MUL_10(anumlength, anumnew, anumnew, aunderflow) ||
                  aoverflow;
        aoverflow=_ADD_DWORD(anumlength, anumnew, astring[i]-_S('0'), anumnew)||
                  aoverflow;
        }
      pointer x;
                                                  //Nachkommastellen berechnen
      x=_NEW(anumlength);
      _INS_BIT(anumlength, 0, 1, x);
      for(i = (dotposinstring+1); i < astringlength; i++)
        {
        aoverflow=_STR_MUL_10(anumlength, anumnew, anumnew, aunderflow) ||
                  aoverflow;
```

```
        aoverflow=_ADD_DWORD(anumlength, anumnew, astring[i]-_S('0'), anumnew) ||
                  aoverflow;
        aoverflow=_STR_MUL_10(anumlength, x, x, aunderflow) || aoverflow;
        }
      aoverflow=_DIV(anumlength, anumdotpos, anumnew, x, anumnew, aunderflow, div0)
                || aoverflow;                                          //Kommakorrektur
      if(kompl) _NOT2(anumlength, anumnew, anumnew);          //Vorzeichen verarbeiten
      _FREE(x);
      }
    }
  else retval = E_INVALIDARG;
  return retval;
  }

API void        FC    _INT          (c_dword        alength,
                                     c_dword        adotpos,
                                     c_pointer      aoriginal,
                                     c_pointer      avorkomma)
  {
  dword d = alength * 32;
  if(adotpos < d)
    _COPYBITS(alength, aoriginal, d-1, adotpos, avorkomma);  //Vorkommastellen kopieren
  else
    zero(avorkomma, d);
  }

API void        FC    _FRAC         (c_dword        alength,
                                     c_dword        adotpos,
                                     c_pointer      aoriginal,
                                     c_pointer      anachkomma)
  {
  if(adotpos > 0)
    _COPYBITS(alength, aoriginal, adotpos-1, 0, anachkomma);  //Nachkommast. kopieren
  else
    zero(anachkomma, alength*4);
  }

API c_boolean   FC    _I_DIV_MOD     (c_dword         alength,
                                      c_dword         adot,
                                      c_pointer       aindata1,
                                      c_pointer       aindata2,
                                      c_pointer       adiv,
                                      c_pointer       amod,
                                      r_boolean       div0)
                                  //ganzzahlige Division mit Restrückgabe
  {
  boolean aoverflow;
  boolean kompl;
  int     s;
  int     outpos;
  int     msb_op1;
  int     msb_op2;
  dword   exactlyshift;
  dword   o;
  pointer op1;
  pointer op2;
  pointer temp;
  boolean z;
  aoverflow = false;
  kompl     = false;
  op1       = _COPY (alength, aindata1);
  op2       = _COPY (alength, aindata2);
  zero(adiv, alength*4);
  zero(amod, alength*4);
  if(_NEGATIVE(alength, op1))
    {
    _NOT2(alength, op1, op1);                                    //Beträge bilden
    kompl = true;
    }
```

```
if(_NEGATIVE(alength, op2))
  {
  _NOT2(alength, op2, op2);
  kompl = !(kompl);
  }
msb_op1 = _MSB(alength, op1);
msb_op2 = _MSB(alength, op2);
div0    = (msb_op2 == NOT_FOUND);
if(!(msb_op1 == NOT_FOUND) && (!div0))                          // x/0 abfangen
   {
  if((_CMP(alength, op1, op2) & CMP_MATCH) != 0)                // x/x abfangen
    {
    _INS_BIT (alength, adot, 1, adiv);
    }
  else
    {
    s            = msb_op1 - msb_op2;                  //Ausgabeposition berechnen
    outpos       = adot + s;
    exactlyshift = (alength * 32) - 2;
    o            = exactlyshift;
    if(msb_op1 > msb_op2)  exactlyshift -= msb_op1;          //Exactlyshift ermitteln
    else                   exactlyshift -= msb_op2;
    if(outpos > (integer)o)
      {
      aoverflow = true;
      }
    else
      {
      dword s1 = exactlyshift;
      dword s2 = exactlyshift;
      if(s <= 0) s1 -= s;
      else       s2 += s;
      _SHL_N(alength, s1, op1, op1);          //Operanden schieben um genau zu rechnen
      _SHL_N(alength, s2, op2, op2);
      z     = true;
      temp  = _NEW(alength);
      while( z && (outpos >= (integer)adot) )          //keine Nachkommastellen bilden
                                                       //Shift & Sub anwenden
        {
        _SUB (alength, op1, op2, temp);
        if(_POSITIVE (alength, temp))
          {
          move(op1, temp, 4*alength);
          _INS_BIT (alength, outpos, true, adiv);
          }
        _SHR_1 (alength, op2, op2);
        z = (!_IZ(alength, op2)) && (!_IZ(alength, op1));
        outpos--;
        }
      _FREE(temp);
      _SHR_N(alength, s1, op1, amod);
      }
    }
  if((!div0) && kompl)
    {
    _NOT2(alength, amod, amod);                    //erwartetes Vorzeichen einbringen
    _NOT2(alength, adiv, adiv);
    }
  }
_FREE (op1);
_FREE (op2);
return aoverflow;
}
```

```
API c_hresult  FC    _NUM2STR          (c_pointer     anum,
                                        c_dword       anumlength,
                                        c_dword       anumdotpos,
                                        c_character   astringdot,
                                        p_CString     astring,
                                        c_dword       astringkommastellen)
                                                              //wandelt Zahl zum String
                  //Betrag der Zahl bilden. Vor- und Nachkommastellen werden voneinander getrennt.
                    //Vorkommastellen /10, den Rest in den String schreiben bis Vorkommastellen = 0.
                       //Höchstwertigste Stelle der Zahl steht rechts im String. Nachschauen ob Zahl
                           //negativ war und evtl. Vorzeichen an String anhängen. String drehen.
                             //Danach Nachkommastellen * 10, Vorkommastelle des Ergebnisses
                                 //wird an String angehangen bis Nachkommastellen = 0.
                               //Desweitern noch Abfragen um String in brauchbares Format
                                 //zu wandeln (z.B. Null vor Komma wenn Zahl = 0.xxx).
  {
  hresult retval = S_OK;
  boolean kompl=false;
  boolean div0=false;
  boolean underflow=false;
  pointer op1;
  pointer op2;
  pointer amod;
  pointer o10;
  dword i;
  dword j;
  dword temp;
  p_CString str=new CString();
  if(astring)
    {
    astring->Clear();
    if(succeeded(retval))
      {
      op1=_COPY(anumlength, anum);
      op2=_NEW(anumlength);
      o10=_NEW(anumlength);                                          //eine "10"
      _INS_BIT(anumlength, anumdotpos+1, 1, o10);
      _INS_BIT(anumlength, anumdotpos+3, 1, o10);
      if(_NEGATIVE(anumlength, op1))
        {
        _NOT2(anumlength, op1, op1);                                 //Betrag bilden
        kompl=true;
        }
      _FRAC(anumlength, anumdotpos, op1, op2);                       //Nach- und
      _INT(anumlength, anumdotpos, op1, op1);                  //Vorkommastellen trennen
      amod=_NEW(anumlength);
      while(!(_IZ(anumlength, op1)))            //Vorkommastellen / 10 bis Vorkommast.=0
        {
        _I_DIV_MOD(anumlength, anumdotpos, op1, o10, op1, amod, div0);
        temp=0;
        for(i=0; i<=3; i++)                  //Umwandeln der Restes in brauchbares Format
          {
          if(_TEST_BIT(anumlength, anumdotpos+i, amod)) temp+=(1<<i);
          }
        astring->AddUnichar((character)(temp+'0'));            //Ziffer in String schreiben
        }
      if(astring->Length()<=0) astring->AddUnichar(_S('0')); //nix im String?, 0 schreiben
      if(kompl) astring->AddUnichar(_S('-'));             //wenn neg. Zahl dann schreib '-'
      astring->Rotate();                                             //String drehen
      astring->AddUnichar(astringdot);                              //Komma schreiben
      for(j=1; ((j<=astringkommastellen) && !_IZ(anumlength, op2)); j++)
                  //Nachkommastellen *10 bis Nachkommastellen=0, Stringkommastellen einhalten
        {
        _STR_MUL_10(anumlength, op2, op2, underflow);
        temp=0;
        for(i=0; i<=3; i++)
                     //Vorkommastellen testen und ermittelte Ziffer in brauchbares Format bringen
          {
```

```
          if(_TEST_BIT(anumlength, anumdotpos+i, op2)) temp+=1<<i;
          }
        astring->AddUnichar((character)(temp+'0'));                    //in String schreiben
        _FRAC(anumlength, anumdotpos, op2, op2);                       //Vorkommastellen weg
        }
      while(astring->GetAt(astring->Length()-1)==_S('0'))       //löscht Nachkommanullen
        astring->Delete(astring->Length()-1, 1);
      if(astring->GetAt(astring->Length()-1)==astringdot)   //Komma löschen wenn keine
                                                                       //Kommastellen
        astring->Delete(astring->Length()-1, 1);
      if(astring->Length()<=0) astring->AddUnichar(_S('0'));           //Null erzeugen
      }
    }
  else retval = E_INVALIDARG;
  return retval;
  }

API c_boolean     FC     _IS_MAX_NEG  (c_dword        alength,
                                       c_pointer      aoriginal)
                                            //liefert true wenn min. neg. Zahl
  {
  boolean boolette=false;
  if(_LSB(alength, aoriginal) == (alength*32)-1)       //wenn lsbpos=msbpos (2er-Kompl.)
    boolette=true;
  return boolette;
  }

API c_boolean     FC     _ABS         (c_dword        alength,
                                       c_pointer      aoriginal,
                                       c_pointer      aabs)
                                                  //Bildet Betrag einer Zahl
  {
  boolean boolette=false;
  if(_NEGATIVE(alength, aoriginal))
    {
    if(_IS_MAX_NEG(alength, aoriginal))       //Betrag der maximalen neg. Zahl nicht möglich
      {
      boolette=true;
      move(aabs, aoriginal, alength*4);          //Betrag der min. neg. Zahl = min. neg. Zahl
      }
    else _NOT2(alength, aoriginal, aabs);                                //Betrag bilden
    }
  return boolette;
  }

__forceinline boolean     FC     _INC           (c_dword        alength,
                                                 c_pointer      ain,
                                                 c_pointer      aout)
                                                       //addiert auf eine Zahl 1
  {
    __asm
    {
        xor       eax,eax                                          //eax auf 0 setzen
        test      ecx,ecx                                                //alength=0?
        jz        ende                                           //ja... sprung ende
        mov       esi,ain                                                //esi:= ain
        mov       edi,aout                                               //edi:= aout
        cld                                                //directionflag:=0 (incdec=4)
        stc                                                             //c-flag:=1
looper:
        lodsd                                                  //eax:= [esi]; esi+4    adc
eax,0                        //eax:=eax+cf
        stosd                                                  //[edi]:= eax; edi+4
        loop      loop                                  //anzahl der 4byte-blöcke (ecx)
                                                               //um 1 verringern
        mov       eax,0                                                  //eax=0
        jno       ende
```

```
          not       eax
ende:
    }
  }

__forceinline boolean     FC     _DEC            (c_dword        alength,
                                                  c_pointer      ain,
                                                  c_pointer      aout,
                                                  r_boolean      aiszero)
                                                       //subtrahiert von einer Zahl 1
  {
   __asm
    {
          xor       eax,eax                                          //eax auf 0 setzen
          mov       ebx,0xffffffff
          test      ecx,ecx                                                //alength=0?
          jz        ende                                            //ja... sprung ende
          mov       esi,ain                                                 //esi:= ain
          mov       edi,aout                                                //edi:= aout
          cld                                                 //directionflag:=0 (incdec=4)
          stc                                                              //c-flag:=1
looper:
          lodsd                                                   //eax:= [esi]; esi+4    sbb
eax,0                     //eax:=eax-cf
          jz        goon
          pushf
          xor       ebx,ebx                                          //prüfen ob Zahl=0
          popf
          jmp       looper2e
goon:
          stosd                                                         //[edi]:= eax; edi+4
          loop      looper              //anzahl der 4byte-blöcke (ecx) um 1 verringern
          jmp       ok
looper2:                                //effektivere Schleife wenn bekannt das Zahl<>0
          lodsd                                                         //eax:= [esi]; esi+4
          sbb       eax,0                                                    //eax:=eax-cf
looper2e:
          stosd                                                   //[edi]:= eax; edi+4    loop
looper2           //anzahl der 4byte-blöcke (ecx) um 1 verringern
ok:
          mov       eax,0                                                          //eax=0
          jno       ende
          not       eax
ende:
          mov       ecx,aiszero
          mov       [ecx],ebx
     }
  }

PI c_boolean    FC     _FACTORIAL      (c_dword        alength,
                                        c_dword        adotpos,
                                        c_pointer      ain,
                                        c_pointer      aout)
                                           //bildet Fakultät einer Zahl
  {
  boolean retval = false;
  pointer op1 = _COPY(alength, ain);
  zero(aout, alength*4);
  boolean b;
  if(_POSITIVE(alength, op1))
    {
    _SHR_N(alength, adotpos, op1, op1);   //Rechtsschieben um _INC, _DEC zu benutzen
    if(!_IZ(alength, op1))
      {
      _INS_BIT(alength, 0, true, aout);                                          //aout=1
      do                                                                  //Fakultät bilden
        {
        retval = _MUL(alength, 0, op1, aout, aout, b);
```

```
        _DEC(alength, op1, op1, b);
        }
      while ((!(b)) && (!(retval)));
      _SHL_N(alength, adotpos, aout, aout);                    //Korrekturschiebung
      }
    else _INS_BIT(alength, adotpos, true, aout);
    }
   else _INS_BIT(alength, adotpos, true, aout);
  _FREE(op1);
  return retval;
  }

API c_boolean   FC    _E_X          (c_dword        alength,
                                     c_dword        adotpos,
                                     c_pointer      ax,
                                     c_pointer      aout)
                                                   //Berechnet e^x, x=b11,b12
  {
  boolean overflow=false;
  boolean kompl=false;
  boolean aunderflow=false;
  boolean div0=false;
  boolean toob=false;
  pointer b11=_COPY(alength,ax);
  pointer b12=_COPY(alength,ax);
  pointer b21;
  pointer b22;
  pointer temp;
  pointer aone;
  zero(aout, alength*4);
  _INS_BIT(alength, adotpos, true, aout);
  if(!(_IZ(alength, b11)))                                          //e^0=1
    {
    if(_NEGATIVE(alength, b11))
      {
      _NOT2(alength, b11, b11);                                  //Betrag op bilden
      _NOT2(alength, b12, b12);
      kompl=true;
      }
    aone=_NEW(alength);
    _INS_BIT(alength, adotpos, 1, aone);
    b21=_COPY(alength, aone);
    b22=_COPY(alength, aone);
    temp=_NEW(alength);
    _ADD(alength, aout, b11, aout);
    do                                                          //Berechnung nach Formel
      {
      overflow=_ADD(alength, aout, temp, aout);
      if(!overflow)
        {
        _ADD(alength, b22, aone, b22);
        toob=_MUL(alength, adotpos, b11, b12, b11, aunderflow);
        if(!toob)
          {
          toob=_MUL(alength, adotpos, b21, b22, b21, aunderflow);
          if(!toob)
            {
            zero(temp, alength*4);
            _DIV(alength, adotpos, b11, b21, temp, aunderflow, div0);
            }
          }
        }
      }
                                                  //Formelbnerechnung wird fortgesetzt bis sich
                                                //das Ergebnis beim aufaddieren nicht mehr ändert
                                                    //so wird i->unendlich simuliert
    while((toob==false) && !(_IZ(alength, temp)) && !overflow);
    if(kompl)
      {                                                         //Vorzeichen op einbringen
```

```
      _DIV(alength, adotpos, aone, aout, aout, aunderflow, div0);
      }
    _FREE(b21);
    _FREE(b22);
    _FREE(temp);
    _FREE(aone);
    }
  _FREE(b11);
  _FREE(b12);
  return overflow;
  }

API c_boolean   FC    _SIN          (c_dword        alength,
                                     c_dword        adotpos,
                                     c_pointer      ain,
                                     c_pointer      aout)
                                                    //Berechbet Sin x, x=b11,b12
  {
  boolean overflow=false;
  boolean kompl=false;
  boolean aunderflow=false;
  boolean div0=false;
  boolean toob=false;
  boolean toggle=false;
  pointer b11=_COPY(alength,ain);
  pointer b12=_COPY(alength,ain);
  pointer b21;
  pointer b22;
  pointer temp;
  pointer aone;
  zero(aout, alength*4);
  if(!(_IZ(alength, b11)))                                          //sin 0 = 0
    {
    if(_NEGATIVE(alength, b11))
      {
      _NOT2(alength, b11, b11);                                     //Betrag vom op bilden
      _NOT2(alength, b12, b12);
      kompl=true;
      }
    aone=_NEW(alength);
    _INS_BIT(alength, adotpos, 1, aone);
    b21=_COPY(alength, aone);
    b22=_COPY(alength, aone);
    temp=_NEW(alength);
    _ADD(alength, aout, ain, aout);
    do                                                              //Berechnung nach Formel
      {
      if(toggle) _NOT2(alength, temp, temp);
      overflow=_ADD(alength, aout, temp, aout);
      toggle=!toggle;
      if(!overflow)
        {
        toob=_MUL(alength, adotpos, b11, b12, b11, aunderflow);
        if(!toob)
          {
          toob=_MUL(alength, adotpos, b11, b12, b11, aunderflow);
          if(!toob)
            {
            _ADD(alength, b22, aone, b22);
            toob=_MUL(alength, adotpos, b21, b22, b21, aunderflow);
            if(!toob)
              {
              _ADD(alength, b22, aone, b22);
              toob=_MUL(alength, adotpos, b21, b22, b21, aunderflow);
              if(!toob)
                {
                zero(temp, alength*4);
                _DIV(alength, adotpos, b11, b21, temp, aunderflow, div0);
                }
```

```
            }
          }
        }
      }
    }
                                          //Schleife läuft bis sich Ergebnis nicht mehr ändert
  while((toob==false) && !(_IZ(alength, temp)) && !overflow);
  if(kompl) _NOT2(alength, aout, aout);              //Vorzeichen vom op einbringen
  _FREE(b21);
  _FREE(b22);
  _FREE(temp);
  _FREE(aone);
  }
_FREE(b11);
_FREE(b12);
return overflow;
}

API c_boolean   FC    _COS          (c_dword        alength,
                                     c_dword        adotpos,
                                     c_pointer      ain,
                                     c_pointer      aout)
                                                    //Berechnet Cos x, x=b12
{
boolean overflow=false;
boolean kompl=false;
boolean aunderflow=false;
boolean div0=false;
boolean toob=false;
boolean toggle=false;
pointer b11;
pointer b12=_COPY(alength,ain);
pointer b21;
pointer b22;
pointer temp;
pointer aone;
zero(aout, alength*4);
_INS_BIT(alength, adotpos, true, aout);
if(!(_IZ(alength, b12)))                                              //cos 0 = 1
  {
  if(_NEGATIVE(alength, b12)) _NOT2(alength, b12, b12);
  aone=_NEW(alength);
  _INS_BIT(alength, adotpos, 1, aone);
  b11=_COPY(alength, aone);
  b21=_COPY(alength, aone);
  b22=_NEW(alength);
  temp=_NEW(alength);
  overflow=_MUL(alength, adotpos, b12, b12, b12, aunderflow);
  if(!overflow)
    do                                                  //Berechnung nach Formel
      {
      if(toggle) _NOT2(alength, temp, temp);
      overflow=_ADD(alength, aout, temp, aout);
      toggle=!toggle;
      if(!overflow)
        {
        toob=_MUL(alength, adotpos, b11, b12, b11, aunderflow);
        if(!toob)
          {
          _ADD(alength, b22, aone, b22);
          toob=_MUL(alength, adotpos, b21, b22, b21, aunderflow);
          if(!toob)
            {
            _ADD(alength, b22, aone, b22);
            toob=_MUL(alength, adotpos, b21, b22, b21, aunderflow);
            if(!toob)
              {
              zero(temp, alength*4);
              _DIV(alength, adotpos, b11, b21, temp, aunderflow, div0);
```

```
                }
              }
            }
          }
        }
                                                    //Schleife läuft bis sich Ergebnis nicht mehr ändert
    while((toob==false) && !(_IZ(alength, temp)) && !overflow);
    _FREE(b21);
    _FREE(b22);
    _FREE(temp);
    _FREE(aone);
    }
  _FREE(b11);
  _FREE(b12);
  return overflow;
  }

API c_boolean   FC    _TAN          (c_dword        alength,
                                     c_dword        adotpos,
                                     c_pointer      ain,
                                     c_pointer      aout,
                                     r_boolean      adiv0)
                                                                    //Berechnet tan x, x=op
  {
  boolean overflow=false;
  boolean aunderflow=false;
  adiv0=false;
  pointer op=_COPY(alength,ain);
  pointer e_sin=_NEW(alength);
  pointer e_cos=_NEW(alength);
  zero(aout, alength*4);
  if(!(_IZ(alength, op)))                                                    //tan 0 = 0
    {
    overflow=_SIN(alength, adotpos, op, e_sin);                  //umsetzen der Formel
    if(!overflow)
      {
      overflow=_COS(alength, adotpos, op, e_cos);
      if(!overflow)
        {
        overflow=_DIV(alength, adotpos, e_sin, e_cos, aout, aunderflow, adiv0);
        }
      }
    }
  _FREE(op);
  _FREE(e_sin);
  _FREE(e_cos);
  return overflow;
  }

API c_boolean   FC    _LN           (c_dword        alength,
                                     c_dword        adot,
                                     c_pointer      ain,
                                     c_pointer      aout,
                                     r_boolean      adiv0)
                                                                    //berechnet ln x, x=in
  {
  boolean overflow = false;
  boolean uf = false;
  pointer xp1 = 0;
  pointer xm1 = 0;
  pointer one = 0;
  pointer two = 0;
  pointer div = 0;
  boolean toob = false;
  pointer in = _COPY(alength, ain);
  zero(aout, 4*alength);
  if(!_NEGATIVE(alength, in) && !_IZ(alength, in))            // ln x, nur für x>0 definiert
    {
    adiv0 = false;
```

```
    one = _NEW(alength);
    _INS_BIT(alength, adot, true, one);
    xm1 = _COPY(alength, in);
    overflow = _SUB(alength, xm1, one, xm1);
    div = _NEW(alength);
    _INS_BIT(alength, adot, true, div);
    two = _NEW(alength);
    xp1 = _NEW(alength);
//  if(_MSB(alength, in) > (adot+1))          //op>2, Berechnung nach Formel
      {
      move(xp1, in, alength*4);
      overflow = _ADD(alength, xp1, one, xp1);
      if(!overflow)
        {
        _DIV(alength, adot, xm1, xp1, xp1, uf, adiv0);
        _MUL(alength, adot, xp1, xp1, xm1, uf);
        _INS_BIT(alength, adot+1, true, two);
        _INS_BIT(alength, adot, false, one);
        move(one, xp1, alength*4);
        do
          {
          overflow = _ADD(alength, aout, one, aout);
          if(!(overflow))
            {
            toob = _MUL(alength, adot, xp1, xm1, xp1, uf);
            if(!(toob))
              {
              toob = _ADD(alength, div, two, div);
              if(!(toob))
                {
                toob = _DIV(alength, adot, xp1, div, one, uf, adiv0);
                }
              }
            }
          }
                                          //Schleife läuft bis sich Ergebnis nicht mehr ändert
        while ((!toob) && (!overflow) && (!_IZ(alength, one)));
        overflow = _ADD(alength, aout, aout, aout);
        }
                                          //Schleife läuft bis sich Ergebnis nicht mehr ändert
      while ((!overflow) && (!toob) && (!_IZ(alength, xp1)));
      }
    }
  else adiv0 = true;
  _FREE(in);
  return overflow;
  }
```

```
boolean     FC    _PI                (c_dword        alength,
                                      c_dword        adotpos,
                                      c_pointer      aout)
                        //liefert Pi in Abhängigkeit von der vom Benutzer angegebenen Genauigkeit
  {
  boolean aunderflow=false;
  boolean div0=false;
  boolean overflow=false;
  boolean toob=false;
  pointer aout_h=_NEW(alength);
  pointer aone=_NEW(alength);
  _INS_BIT(alength, adotpos,   true, aone);
  pointer atwo=_NEW(alength);
  _INS_BIT(alength, adotpos+1, true, atwo);
  pointer afour=_NEW(alength);
  _INS_BIT(alength, adotpos+2, true, afour);
  pointer afive=_NEW(alength);
  _INS_BIT(alength, adotpos,   true, afive);
  _INS_BIT(alength, adotpos+2, true, afive);
  pointer asix=_NEW(alength);
  _INS_BIT(alength, adotpos+1, true, asix);
  _INS_BIT(alength, adotpos+2, true, asix);
  pointer a=_NEW(alength);
  _INS_BIT(alength, adotpos+2, true, a);
  pointer a_h=_NEW(alength);
  pointer b=_NEW(alength);
  pointer b_h=_NEW(alength);
  pointer c=_NEW(alength);
  pointer c_h=_NEW(alength);
  pointer d=_NEW(alength);
  pointer d_h=_NEW(alength);
  pointer temp1=_NEW(alength);
  pointer temp2=_NEW(alength);
  pointer m1=_NEW(alength);
  pointer m1_16=_NEW(alength);
  _INS_BIT(alength, adotpos+4, true, m1_16);
  pointer m1_16i=_NEW(alength);
  _INS_BIT(alength, adotpos,   true, m1_16i);
  pointer m2=_NEW(alength);
  pointer m2_8=_NEW(alength);
  _INS_BIT(alength, adotpos+3, true, m2_8);
  pointer m2_8i=_NEW(alength);
  zero(aout, alength*4);
  _DIV(alength, adotpos, aone, atwo, b, aunderflow, div0);        //umsetzen der Formel
  _DIV(alength, adotpos, aone, afive, c, aunderflow, div0);
  _DIV(alength, adotpos, aone, asix, d, aunderflow, div0);
  _SUB(alength, a, b, temp1);
  _ADD(alength, c, d, temp2);
  _SUB(alength, temp1, temp2, aout_h);
  do
    {
    overflow=_ADD(alength, aout, aout_h, aout);
    if(!toob)
      {
      toob=_MUL(alength, adotpos, m1_16i, m1_16, m1_16i, aunderflow);
      if(!toob)
        {
        _DIV(alength, adotpos, aone, m1_16i, m1, aunderflow, div0);
        toob=_IZ(alength, m1);
        if(!toob)
          {
          _ADD(alength, m2_8i, m2_8, m2_8i);
          _ADD(alength, m2_8i, aone, a_h);
          _ADD(alength, m2_8i, afour, b_h);
          _ADD(alength, m2_8i, afive, c_h);
          _ADD(alength, m2_8i, asix, d_h);
          _DIV(alength, adotpos, afour, a_h, a, aunderflow, div0);
          _DIV(alength, adotpos, atwo, b_h, b, aunderflow, div0);
          _DIV(alength, adotpos, aone, c_h, c, aunderflow, div0);
```

```
          _DIV(alength, adotpos, aone, d_h, d, aunderflow, div0);
          _SUB(alength, a, b, temp1);
          _ADD(alength, c, d, temp2);
          _SUB(alength, temp1, temp2, m2);
          toob=_IZ (alength, m2);
          if(!toob)
            {
            toob=_MUL(alength, adotpos, m1, m2, aout_h, aunderflow);
            }
          }
        }
      }
    }
                                                    //Schleife läuft bis sich Ergebnis nicht mehr ändert
  while(!toob && !overflow);
  _FREE(aout_h);
  _FREE(aone);
  _FREE(atwo);
  _FREE(afour);
  _FREE(afive);
  _FREE(asix);
  _FREE(a);
  _FREE(a_h);
  _FREE(b);
  _FREE(b_h);
  _FREE(c);
  _FREE(c_h);
  _FREE(d);
  _FREE(d_h);
  _FREE(temp1);
  _FREE(temp2);
  _FREE(m1);
  _FREE(m1_16);
  _FREE(m1_16i);
  _FREE(m2);
  _FREE(m2_8);
  _FREE(m2_8i);
  return overflow;
  }

API c_boolean   FC    _ARCSIN         (c_dword        alength,
                                       c_dword        adotpos,
                                       c_pointer      pi_2,
                                       c_pointer      ain,
                                       c_pointer      aout)
                                                    //berechnen arcsin x, x=op
  {
  boolean overflow=false;
  boolean aunderflow=false;
  boolean toob=false;
  boolean div0=false;
  boolean kompl=false;
  pointer a1;
  pointer a2;
  pointer b1;
  pointer b2;
  pointer c1;
  pointer c2;
  pointer aone;
  pointer atwo;
  pointer temp;
  pointer div1;
  pointer div2;
  pointer op=_COPY(alength, ain);
  zero(aout, alength*4);
  if(!_IZ(alength, op))                                                    //arcsin 0
    {
    aone=_NEW(alength);
    _INS_BIT(alength, adotpos, 1, aone);
```

```
if(_NEGATIVE(alength, op))
  {
  _NOT2(alength, op, op);                                    //Betrag op bilden
  kompl=true;
  }
if(_CMP(alength, op, aone) == CMP_EQUAL)                     //arcsin 1
  {
  move(aout, pi_2, alength*4);
  }
else
  {
  if(_CMP(alength, op, aone) == CMP_BIGGER)                  //arcsin x mit x>1 n.d.
    {
    overflow=true;
    }
  else
    {
    atwo=_NEW(alength);
    _INS_BIT(alength, adotpos+1, true, atwo);
    temp=_NEW(alength);
    a1=_COPY(alength, op);
    a2=_NEW(alength);
    _MUL(alength, adotpos, a1, a1, a2, aunderflow);
    b1=_COPY(alength, aone);
    b2=_COPY(alength, aone);
    c1=_COPY(alength, atwo);
    c2=_NEW(alength);
    _INS_BIT(alength, adotpos, true, c2);
    _INS_BIT(alength, adotpos+1, true, c2);
    div1=_NEW(alength);
    div2=_NEW(alength);
    move(aout, op, alength*4);                               //Berechnung nach Formel
    do
      {
      toob=_MUL(alength, adotpos, c1, c2, div2, aunderflow);
      if(!toob)
        {
        toob=_MUL(alength, adotpos, b1, b2, b1, aunderflow);
        if(!toob)
          {
          toob=_MUL(alength, adotpos, a1, a2, a1, aunderflow);
          if(!toob)
            {
            toob=_MUL(alength, adotpos, a1, b1, div1, aunderflow);
            if(!toob)
              {
              toob=_DIV(alength, adotpos, div1, div2, temp, aunderflow, div0);
              toob=_IZ(alength, temp) || toob;
              if(!toob)
                {
                overflow=_ADD(alength, aout, temp, aout);
                if(!overflow)
                  {
                  toob=_ADD(alength, b2, atwo, b2);
                  toob=_ADD(alength, c2, aone, c2) || toob;
                  if(!toob)
                    {
                    toob=_MUL(alength, adotpos, c1, c2, c1, aunderflow);
                    toob=_ADD(alength, c2, aone, c2);
                    }
                  }
                }
              }
            }
          }
        }
      }
                                                             //Schleife läuft bis sich Ergebnis nicht mehr ändert
    while(!toob && !overflow);
```

```
        _FREE(atwo);
        _FREE(a1);
        _FREE(a2);
        _FREE(b1);
        _FREE(b2);
        _FREE(c1);
        _FREE(c2);
        _FREE(div1);
        _FREE(div2);
        }
      }
    if(!overflow && kompl) _NOT2(alength, aout, aout);        //Vorzeichen op einbringen
    _FREE(aone);
    }
  _FREE(op);
  return overflow;
  }

API c_boolean   FC    _ARCCOS         (c_dword        alength,
                                       c_dword        adotpos,
                                       c_pointer      pi,
                                       c_pointer      pi_2,
                                       c_pointer      ain,
                                       c_pointer      aout)
                                                      //berechnen arccos x, x=op
  {
  boolean overflow=false;
  boolean aunderflow=false;
  boolean toob=false;
  boolean div0=false;
  boolean kompl=false;
  pointer a1;
  pointer a2;
  pointer b1;
  pointer b2;
  pointer c1;
  pointer c2;
  pointer aone;
  pointer atwo;
  pointer temp;
  pointer div1;
  pointer div2;
  pointer op=_COPY(alength, ain);
  zero(aout, alength*4);
  if(_IZ(alength, op))                                    //arccos 0
    {
    move(aout, pi_2, alength*4);
    }
  else
    {
    if(_NEGATIVE(alength, op))
      {
      _NOT2(alength, op, op);                             //Betrag op bilden
      kompl=true;
      }
    aone=_NEW(alength);
    _INS_BIT(alength, adotpos, 1, aone);
    if(_CMP(alength, op, aone) == CMP_EQUAL)              //arccos 1
      {
      if(kompl) move(aout, pi, alength*4);
      }
    else
      {
      if(_CMP(alength, op, aone) == CMP_BIGGER)           //arccos x mit x>1 n.d.
        {
        overflow=true;
        }
      else
        {
```

```
atwo=_NEW(alength);
_INS_BIT(alength, adotpos+1, true, atwo);
temp=_NEW(alength);
a1=_COPY(alength, op);
a2=_NEW(alength);
_MUL(alength, adotpos, a1, a1, a2, aunderflow);
b1=_COPY(alength, aone);
b2=_COPY(alength, aone);
c1=_COPY(alength, atwo);
c2=_NEW(alength);
_INS_BIT(alength, adotpos, true, c2);
_INS_BIT(alength, adotpos+1, true, c2);
div1=_NEW(alength);
div2=_NEW(alength);
move(aout, op, alength*4);                                  //Berechnung nach Formel
do
  {
  toob=_MUL(alength, adotpos, c1, c2, div2, aunderflow);
  if(!toob)
    {
    toob=_MUL(alength, adotpos, b1, b2, b1, aunderflow);
    if(!toob)
      {
      toob=_MUL(alength, adotpos, a1, a2, a1, aunderflow);
      if(!toob)
        {
        toob=_MUL(alength, adotpos, a1, b1, div1, aunderflow);
        if(!toob)
          {
          toob=_DIV(alength, adotpos, div1, div2, temp, aunderflow, div0);
          toob=_IZ(alength, temp) || toob;
          if(!toob)
            {
            overflow=_ADD(alength, aout, temp, aout);
            if(!overflow)
              {
              toob=_ADD(alength, b2, atwo, b2);
              toob=_ADD(alength, c2, aone, c2) || toob;
              if(!toob)
                {
                toob=_MUL(alength, adotpos, c1, c2, c1, aunderflow);
                toob=_ADD(alength, c2, aone, c2);
                }
              }
            }
          }
        }
      }
    }
  }
                                        //Schleife läuft bis sich Ergebnis nicht mehr ändert
while(!toob && !overflow);
if(!overflow)                           //"Rest"rechnung und Vorzeichen einbringen
  {
  overflow=_SUB(alength, pi_2, aout, aout);
  if(kompl)
    {
    _NOT2(alength, aout, aout);
    _ADD(alength, aout, pi, aout);
    }
  }
_FREE(temp);
_FREE(atwo);
_FREE(a1);
_FREE(a2);
_FREE(b1);
_FREE(b2);
_FREE(c1);
_FREE(c2);
```

```
        _FREE(div1);
        _FREE(div2);
        }
      }
    _FREE(aone);
    }
  _FREE(op);
  return overflow;
  }

API c_boolean   FC    _ARCTAN          (c_dword        alength,
                                        c_dword        adotpos,
                                        c_pointer      pi_2,
                                        c_pointer      pi_4,
                                        c_pointer      ain,
                                        c_pointer      aout)
                                                        //berechnen arctan x, x=op
  {
  boolean overflow=false;
  boolean kompl=false;
  boolean aunderflow=false;
  boolean toggle=false;
  boolean toob=false;
  boolean div0=false;
  pointer op=_COPY(alength, ain);
  pointer aout_h;
  pointer aone;
  pointer atwo;
  pointer xpot;
  pointer xpot_h;
  pointer gaza;
  pointer gaza_h;
  pointer m_erg;
  zero(aout, alength*4);
  if(!_IZ(alength, op))                                          //arctan 0 = 0
    {
    if(_NEGATIVE(alength, op))
      {
      _NOT2(alength, op, op);                                    //Betrag von op bilden
      kompl=true;
      }
    aone=_NEW(alength);
    _INS_BIT(alength, adotpos, true, aone);
    if(_CMP(alength, op, aone) == CMP_EQUAL)                     //arctan 1
      {
      move(aout, pi_4, alength*4);
      }
    else
      {
      atwo=_NEW(alength);
      _INS_BIT(alength, adotpos+1, true, atwo);
      xpot=_COPY(alength, op);
      xpot_h=_NEW(alength);
      _MUL(alength, adotpos, xpot, xpot, xpot_h, aunderflow);
      gaza=_COPY(alength, aone);
      gaza_h=_COPY(alength, atwo);
      if(_CMP(alength, op, aone) == CMP_SMALLER)        //op<1 .. Berechnung nach Formel
        {
        aout_h=_COPY(alength, op);
        while(!toob && !overflow)          //Schleife läuft bis sich Ergebnis nicht mehr ändert

          {
          if(toggle) _NOT2(alength, aout_h, aout_h);
          toggle=!toggle;
          overflow=_ADD(alength, aout, aout_h, aout);
          if(!overflow)
            {
            _MUL(alength, adotpos, xpot, xpot_h, xpot, aunderflow);
            toob=_ADD(alength, gaza, gaza_h, gaza);
```

```
          if(!toob)
            {
            _DIV(alength, adotpos, xpot, gaza, aout_h, aunderflow, div0);
            toob=_IZ(alength, aout_h);
            }
          }
        }
      _FREE(aout_h);
      }
    else                                                    //op>1 .. Berechnung nach Formel
      {
      m_erg=_NEW(alength);
      move(aout, pi_2, alength*4);
      aout_h=_NEW(alength);
      _DIV(alength, adotpos, aone, op, aout_h, aunderflow, div0);
      while(!toob && !overflow)          //Schleife läuft bis sich Ergebnis nicht mehr ändert
        {
        toggle=!toggle;
        if(toggle) _NOT2(alength, aout_h, aout_h);
        overflow=_ADD(alength, aout, aout_h, aout);
        if(!overflow)
          {
          toob=_MUL(alength, adotpos, xpot, xpot_h, xpot, aunderflow);
          if(!toob)
            {
            _ADD(alength, gaza, gaza_h, gaza);
            toob=_MUL(alength, adotpos, xpot, gaza, m_erg, aunderflow);
            if(!toob)
              {
              _DIV(alength, adotpos, aone, m_erg, aout_h, aunderflow, div0);
              toob=_IZ(alength, aout_h);
              }
            }
          }
        }
      _FREE(aout_h);
      _FREE(m_erg);
      }
    _FREE(atwo);
    _FREE(xpot);
    _FREE(xpot_h);
    _FREE(gaza);
    _FREE(gaza_h);
    }
  if(kompl) _NOT2(alength, aout, aout);                  //Vorzeichen von op einbringen
  _FREE(aone);
  }
_FREE(op);
return overflow;
}

API c_boolean   FC    _WURZEL          (c_dword        alength,
                                        c_dword        adotpos,
                                        c_pointer      azahl,
                                        c_pointer      awurzel,
                                        c_pointer      aout,
                                        r_boolean      adiv0)
                                                  //xte Wurzel aus y berechnen, y=op1, x=op2
{
boolean overflow=false;
boolean aunderflow=false;
boolean kompl=false;
pointer op1=_COPY(alength, azahl);
pointer op2=_COPY(alength, awurzel);
pointer aone=_NEW(alength);
_INS_BIT(alength, adotpos, true, aone);
zero(aout, alength*4);
if(!_IZ(alength, op2))                                  //die 0-te Wurzel ist nicht definiert
  {
```

```
    if(!_IZ(alength, op1))                                          //Wurzel aus Null
      {
      if(!_NEGATIVE(alength, op1))                                  //Wurzel aus neg. Zahl n.d.
        {
        if(_NEGATIVE(alength, op2))
          {
          _NOT2(alength, op2, op2);
          kompl=true;
          }
        overflow=_LN(alength, adotpos, op1, op1, adiv0);            //Berechnung nch Formel
        if((!overflow) && (!adiv0))
          {
          overflow=_DIV(alength, adotpos, aone, op2, op2, aunderflow, adiv0);
          if(!overflow)
            {
            overflow=_MUL(alength, adotpos, op1, op2, op2, aunderflow);
            if(!overflow)
              {
              overflow=_E_X(alength, adotpos, op2, aout);
              if(!overflow && kompl)                                //Korrektur bei neg. Wurzel mit 1/Erg.
                {
                _DIV(alength, adotpos, aone, aout, aout, aunderflow, adiv0);
                }
              }
            }
          }
        else overflow=true;
        }
      else
        {
        if(_NEGATIVE(alength, op2)) overflow=true;                  //negative Wurzel aus Null n.d.
        }
      }
    else overflow=true;
    _FREE(aone);
    _FREE(op1);
    _FREE(op2);
    return overflow;
    }

API c_boolean   FC    _POTENZ          (c_dword        alength,
                                        c_dword        adotpos,
                                        c_pointer      azahl,
                                        c_pointer      aexponent,
                                        c_pointer      aout,
                                        r_boolean      adiv0)
                                                        //x^y berechnen, x=op1, y=op2
    {
    boolean overflow=false;
    boolean aunderflow=false;
    boolean kompl=false;
    boolean gotoend=false;
    pointer op1=_COPY(alength, azahl);
    pointer op2=_COPY(alength, aexponent);
    pointer temp=_NEW(alength);
    pointer atwo=_NEW(alength);
    _INS_BIT(alength, adotpos+1, true, atwo);
    zero(aout, alength*4);
    if(!_IZ(alength, op1))                                          //wenn x=0 => Ergebnis=0
      {
      _INS_BIT(alength, adotpos, true, aout);
      if(!_IZ(alength, op2))                                        //wenn y=0 => Ergebnis=1
        {
        if(_NEGATIVE(alength, op1))                                 //wenn x negativ ..
          {
          _FRAC(alength, adotpos, op2, temp);
          if(_IZ(alength, temp))                                    //..dann muß y ganzzahlig sein
            {
```

```
        _NOT2(alength, op1, op1);
        _I_DIV_MOD(alength, adotpos, op2, atwo, atwo, temp, adiv0);
        if(!_IZ(alength, temp)) kompl=true;
        }
      else
        {
        gotoend=true;
        overflow=true;
        }
      }
    if(!gotoend)
      {
      overflow=_LN(alength, adotpos, op1, op1, adiv0);       //Berechnung nach Formel
      if((!overflow) && (!adiv0))
        {
        overflow=_MUL(alength, adotpos, op1, op2, op2, aunderflow);
        if(!overflow)
          {
          overflow=_E_X(alength, adotpos, op2, aout);
          }
        }
      if(kompl) _NOT2(alength, aout, aout);
      }
    }
  }
else
  {                                                                    //0^-x und 0^0 n.d.
  if(_NEGATIVE(alength, op2) || _IZ(alength, op2)) overflow=true;
  }
_FREE(atwo);
_FREE(temp);
_FREE(op1);
_FREE(op2);
return overflow;
}
```

1.2.1.4.2 Kode CReal.cpp

Der Kode von CReal ist lediglich ein Wrapper für den von RealFunctions, und daher hier nicht von interesse.

1.2.1.5 Modul XML-Parser

1.2.1.5.1 Kode CXMLBase.cpp

```
#include        "XML.h"

//Konstruktor erstellt m_string
CXMLBase::CXMLBase          (c_boolean athreadsecured) : CBase(athreadsecured)
  {
  m_string = new CString(false);
  }


//Destruktor löscht m_string
CXMLBase::~CXMLBase         (void)
  {
  SET_NULL(m_string)
  }


c_hresult     SC    CXMLBase::FromString          (cp_CString astring)
  {
  hresult retval = S_OK;
  if(astring)
    {
//Bei eventuell gültiger Eingabe Löschen vorheriger Daten
    retval = Clear();
    }
//Sonst Fehler
  else retval = E_INVALIDARG;
  return retval;
  }


c_dword       SC    CXMLBase::ToString            (cp_CString astring)
  {
  hresult retval = S_OK;
  if(astring)
    {
//Löschen vorheriger Daten in astring
    retval = astring->Clear();
    }
//Sonst Fehler
  else retval = E_INVALIDARG;
  return retval;
  }


c_hresult     SC    CXMLBase::Clear               (void)
  {
//Löschen des Offsets
  m_offset = 0;
//Löschen der String-Daten
  return m_string->Clear();
  }


c_dword       SC    CXMLBase::Type                (void)
  {
//Rückgabe des Code-Wertes für ein XML Grundobjekt
//(sollte es im Projekt eigentlich nicht geben)
  return XML_BASE;
  }
```

```
c_hresult      SC    CXMLBase::Get_Text              (cp_CString astring)
  {
//Rückgabe des Textes eines XML-Objekts
  return astring->FromStr(m_string);
  }


c_hresult      SC    CXMLBase::Set_Text              (cp_CString astring)
  {
//Schreiben des Textes eines XML-Objekts
  return m_string->FromStr(astring);
  }
```

1.2.1.5.2 Kode XMLFunction.cpp

```
#include         "XML.h"

API c_hresult SC xml_type(p_CString astr, rp_CXMLBase abase)
  {
//Variable für XML-Knotenobjekt
  p_CXMLNode node = 0;
//Variable für XML-Prozessanweisungsobjekt
  p_CXMLPI pi = 0;
  hresult retval = S_OK;

//Löschen von abase
  abase = 0;

//Erstellen der Variable pi
  pi = new CXMLPI(false);

//Test mittels FromString ob astr ein Prozessanweisungsobjekt ist
//Wenn astr Prozessanweisungsobjekt ist, Schreiben von pi auf abase
  if(pi->FromString(astr) == S_OK)
    {
    abase = pi;
    }
//Sonst Löschen von pi
  else
    {
    SET_NULL(pi)
//Erstellen der Vaiable node
    node = new CXMLNode(false);
//Test mittels FromString ob astr ein Knotenobjekt ist
//Wenn astr Knotenobjekt ist, Schreiben von node auf abase
    if (node->FromString(astr) == S_OK)
      {
      abase = node;
      }
//Sonst Löschen von node
//Setzen von retval auf S_FALSE, da in diesem Fall für das Programm nicht verwertbare Daten vorliegen
    else
      {
      SET_NULL(node)
      retval = S_FALSE;
      }
    }


  return retval;
  }
```

1.2.1.5.3 Kode CXMLPI.cpp

```
#include        "XML.h"


//Konstruktor
CXMLPI::CXMLPI          (c_boolean athreadsecured) : CXMLBase(athreadsecured)
  {
  }

//Destruktor
CXMLPI::~CXMLPI         (void)
  {
  }


c_hresult     SC    CXMLPI::FromString           (cp_CString astring)
  {
//Indexvariable
  dword i=0;
//Variable für aktuelles Zeichen
  character x;


//Aufrufen der FromString Funktion vom CXMLBase Objekt
  hresult retval = CXMLBase::FromString(astring);

//Wenn o.g. Aufruf erfolgreich
  if(succeeded(retval))
    {
//Wenn 1.Zeichen gleich '<' und 2.Zeichen gleich '?' ist
//Also Anfang einer Prozessanweisung
        if ((astring->GetAt(0)=='<') && (astring->GetAt(1)=='?'))
      {
//Löschen der beiden Zeichen
               astring->Delete(0,2);

//Solange i-tes Zeichhen nicht '?' und (i+1)-tes Zeichen nicht '>' ist
//Also Ende der Prozessanweisung
       while ( ! (( (x=astring->GetAt(i)) == _S('?')) &&
                    (astring->GetAt(i+1) == _S('>')) ))
         {
//Wenn i-tes Zeichen ´'´ oder ´"´ ist überspringen dieses Teils bis zum
//zugehörigen ´'´oder ´"´ Zeichen
//Bestimmung des zugehörigen schließenden Zeichens mittels QuoteEnd
         if((x == _S('\'')) || (x == _S('\"')))
           {
           i=astring->QuoteEnd(i)+1;
           }
//Sonst i inkrementieren, um nächstes Zeichen zu prüfen
                     else i++;
         }
//Wenn i>0 (also wenn zwischen Anfang und Ende Text vorhanden ist) kopiere
//i Zeichen (also genau der Text) nach m_string
        if (i>0) m_string->AddStr(astring,i);
//Löschen des Textes und des Endes der Prozessanweisung
        astring->Delete(0,i+2);
               }
//Setzen von retval auf S_FALSE, da in diesem Fall für das Programm nicht verwertbare Daten vorliegen
    else retval = S_FALSE;
        }

  return retval;
  }
```

```
c_dword        SC    CXMLPI::ToString              (cp_CString astring)
  {

//Aufrufen der ToString Funktion vom CXMLBase Objekt
  hresult retval = CXMLBase::ToString(astring);
//Wenn o.g. Aufruf erfolgreich
  if(succeeded(retval))
    {
//Hinzufügen der Zeichen '<' und '?' (also den Anfang der Prozessanweisung) zu astring
    astring->AddUnichar(_S('<'));
    astring->AddUnichar(_S('?'));
//Hinzufügen von m_string (also der Text der Prozessanweisung) zu astring
    astring->AddStr(m_string);
//Hinzufügen der Zeichen '?' und '>' (also den Anfang der Prozessanweisung) zu astring
    astring->AddUnichar(_S('?'));
    astring->AddUnichar(_S('>'));
    }
  return retval;
  }



c_dword        SC    CXMLPI::Type                  (void)
  {
//Rückgabe des Code-Wertes für ein XML-Prozessanweisungsobjekt
  return XML_PROCESSING_INSTRUCTION;
  }
```

1.2.1.5.4 Kode CXMLNode.cpp

```
#include        "XML.h"


CXMLNode::CXMLNode          (c_boolean athreadsecured) : CXMLBase(athreadsecured)
  {
//Konstruktor erstellt m_name, m_subnodes und m_attributes
  m_name = new CString(false);
  m_subnodes = new CObjectList(false);
  m_attributes = new CDoubleStringList(false);
  }


CXMLNode::~CXMLNode         (void)
  {
//Destruktor löscht m_name, m_subnodes und m_attributes
  SET_NULL(m_name)
  SET_NULL(m_subnodes)
  SET_NULL(m_attributes)
  }



c_hresult     SC    CXMLNode::FromString          (cp_CString astring)
  {
//Indexvariablen
  dword i=1,j;
//Variable für aktuelles Zeichen
  character x;
//Variable für Stringlänge
  dword str_len;
//Variable (Zeiger auf Stringobjekt) für Endtag
  p_CString tag_end;
//Variable für Offset
  dword offset=0;
//Variable (Zeiger auf CXMLBase Objekt) für Unterknoten
  p_CXMLBase abase;
//Variable (Zeiger auf Stringobjekt) für Attributname
  p_CString att_name;
//Variable (Zeiger auf Stringobjekt) für Attributwert
  p_CString att_value;
//Variable (Boolean) ob Endtag gefunden
  boolean found=false;


//Aufrufen der FromString Funktion vom CXMLBase Objekt
  hresult retval = CXMLBase::FromString(astring);
//Wenn o.g. Aufruf erfolgreich
  if(succeeded(retval))
    {
//str_len auf Länge von astring gesetzt
    str_len = astring->Length();
//Wenn 1.Zeichen gleich '<'ist
    if(astring->GetAt(0)=='<')
      {
//Solange i = Stringlänge und x (also i-tes Zeichen) > ' ' (also kein Steuerzeichen)
//und x ? '>' (also Endezeichen des öffnenden Tags) und x ? '/' (also Tag eines leeren Knotens) ist
      while ((i<=str_len)&&(((x=astring->GetAt(i)) > _S(' ')) &&
             (x != _S('>')) && (x != _S('/'))))
        {
//Hinzufügen von x zu m_name
        m_name->AddUnichar(x);
//i inkrementieren, um nächstes Zeichen zu prüfen
        i++;
        }
      }
```

```
//Setzen von retval auf S_FALSE, da in diesem Fall für das Programm nicht verwertbare Daten vorliegen
    else retval=S_FALSE;

//Solange i-tes Zeichen gleich ‘ ‘ ist
    while (astring->GetAt(i) == _S(' '))
      {
//Solange i = Stringlänge und x (also i-tes Zeichen) gleich  ’ ’ ist
      while ((i<=str_len) && ((x=astring->GetAt(i)) == _S(' ')))
//i inkrementieren (also Leerzeichen überspringen)
        i++;
//Wenn x ? ‘/’ und x ? ‘>’ (also kein Endezeichen)
      if (((x=astring->GetAt(i)) != _S('/')) && (x != _S('>')))
        {
//Erstellen der Variablen att_name und att_value
        att_name = new CString(false);
        att_value = new CString(false);
//Solange i = Stringlänge und x ? ‘=’ ist
        while ((i<=str_len) && ((x=astring->GetAt(i)) != _S('=')))
          {
          i++;
//x (also aktuelles Zeichen) wird att_name zugefügt
//folglich befindet sich in att_name nach der Schleife der Attributname
          att_name->AddUnichar(x);
          }
//mittels QuoteEnd wird die Position des zum Attributwert-Anfangszeichen zugehörige Endzeichen gesucht
//Dieser Wert wird j zugewiesen
        j=((astring->QuoteEnd(i+1))-1);
//i um 2 erhöhen, um Leerzeichen und Anfnagszeichen zu überspringen
//Aktuelles Zeichen wird att_value zugefügt
//folglich befindet sich in att_value nach der Schleife der Attributwert
        for ((i=i+2);(i<=j);(i++))
          att_value->AddUnichar(astring->GetAt(i));

//m_attributes wird aktueller Attributname und zugehöriger Attributwert zugefügt
        m_attributes->Add(att_name, att_value);
//Löschen von att_name und att_value
        SET_NULL(att_name);
        SET_NULL(att_value);
        i+=1;
        }
      }

//Wenn “/>” im String enthalten ist (also leerer Tag)
    if (astring->IsAtUnicode(_S("/>"),i))
      {
//Lösche Tag und Tagendezeichen
      retval=astring->Delete(0,(i+2));
      }
//Sonst normaler Knoten
    else
      {


//Wenn i-tes Zeichen gleich ‘>’
      if ((astring->GetAt(i)) == '>')
        {
//Lösche Tag und Tagendezeichen
        astring->Delete(0,i+1);
//Erstellen der Variable tag_end
        tag_end = new CString(false);
//In tag_end wird der schließende Tag erstellt
        tag_end->AddUnicode(_S("</"));
        tag_end->AddStr(m_name);
        tag_end->AddUnicode(_S(">"));


//Solange str_len (Stringlänge) ? 0 (also String nicht leer) und nicht found
        while(((str_len=(astring->Length())) != 0) && !(found))
          {
```

```
            i=0;
//Solange (i+1) = str_len und x (aktuelles Zeichen) ? '<'
            while (((i+1)<=str_len) && ((x=astring->GetAt(i)) != _S('<')))
//i inkrementieren (also alle Zeichen bis '<' überspringen)
              i++;

//Wenn i>0 (also wenn Text zwischen öffnendem und schließendem Tag vorhanden ist)
            if(i>0)
              {
//Offset um i (also um Länge des Textes) erhöhen
              offset+=i;
//i Zeichen (also genau der Text) von astring zu m_string hinzufügen
              m_string->AddStr(astring,i);
//i Zeichen (also wieder der Text) von astring Löschen
              astring->Delete(0,i);
              }

//Wenn x gleich '<'
            if (x == '<')
              {
//Wenn tag_end in astring an Position 0 vorhanden ist
              if (astring->IsAtStr(tag_end,0))
                {
//Löschen von Länge von tag_end Zeichen (also im Endeffekt genau tag_end)
                astring->Delete(0,tag_end->Length());
//found wird auf true gesetzt, da schließender Tag schon gefunden wurde
                found = true;
                }
              else
                {
                abase=0;
//Aufrufen der Funktion xml_type mit den Parametern astring (also der Reststring)
//und abase (XMLBase Objekt für Unterknoten)
//Die Funktion bestimmt, ob Unterknoten vorhanden sind, indem sie rekursiv die verschiedenen FromString
Routinen der Objekte aufruft
//und vorhandene Unterknoten in abase zurückgibt
                retval=(xml_type(astring,abase));
//Wenn Aufruf erfolgreich
                if (retval == S_OK)
                  {
//m_offset des abase Objekts (also des Unterknotens) wird auf den Wert von offset gesetzt
                  abase->m_offset=offset;
//m_subnodes wird der Unterknoten abase zugefügt
                  m_subnodes->Add(abase);
//Löschen von abase
                  SET_NULL(abase)
                  }
                else
                  {
//Sonst
//Angängen von x an m_string
                  m_string->AddUnichar(x);
//Löschen des aktuellen Zeichens
                  astring->Delete(0,1);
                  }
                }
              }
//Prüfung für Rückgabewert
          if(succeeded(retval) && !(found)) retval= S_FALSE;
          }
        else
//Setzen von retval auf S_FALSE, da in diesem Fall für das Programm nicht verwertbare Daten vorliegen
          retval = S_FALSE;
        }
      }
    return retval;
    }
```

```
c_dword       SC    CXMLNode::ToString             (cp_CString astring)
  {
//Indexvariablen
  dword i, cnt=0;
//Variable für Offset
  dword offset=0;
//Variable für aktuelles Zeichen
  character x;
//Variable (Zeiger auf Stringoblekt) für Text zwischen öffnendem und schließendem Tag
  p_CString text_between;
//Variable (Zeiger auf Stringobjekt) für einzufügenden Text
  p_CString insert_string;
//Variable (Zeiger auf CXMLBase Objekt) für Unterknoten
  p_CXMLBase abase=0;
// Variable (Zeiger auf Stringobjekt) für Attributname
  p_CString att_name;
// Variable (Zeiger auf Stringobjekt) für Attributwert
  p_CString att_value;



//Aufrufen der ToString Funktion vom CXMLBase Objekt
  hresult retval = CXMLBase::ToString(astring);
//wenn o.g. Aufruf erfolgreich
  if(succeeded(retval))
    {
//Erstellen der Variablen text_between und insert_string
    text_between = new CString(false);
    insert_string = new CString(false);

//Einfügen von '<' und m_name zu astring (Erstellung von öffnendem Tag)
    astring->AddUnichar(_S('<'));
    astring->AddStr(m_name);

//Einfügen der verschiedenen Attribute
    for (i=1;i <= m_attributes->Length();i++)
      {
//Erstellen der Variablen att_name und att_value
      att_name = new CString(false);
      att_value = new CString(false);
//Attribut (Name und Wert) von m_attributes nach att_name und att_value kopieren
      m_attributes->GetAt(i-1,att_name,att_value);
      astring->AddUnichar(_S(' '));
//Einfügen von att_name (Attributname) zu astring
      astring->AddStr(att_name);
      astring->AddUnichar(_S('='));

//Solange cnt < Länge von att_value und x (aktuelles Zeichen) ungleich "'" und ungleich '"'
      while ((cnt<att_value->Length()) &&
            ((x=att_value->GetAt(cnt))!=_S('\''))&&(x!=_S('"')))
        cnt++;
//Wenn x gleich "'" setze x auf '"'
      if(x==_S('\'')) x=_S('"');
//Sonst setze x auf "'"
      else x=_S('\'');
//Einfügen von att_value (Attributwert) zu astring
      astring->AddUnichar(x);
      astring->AddStr(att_value);
      astring->AddUnichar(x);
      }
//Kopieren von m_string nach text_between
    text_between->AddStr(m_string);
//Wenn kein Text vorhanden und keine Unterknoten vorhanden Einfügen von "/>" zu astring
//Damit wird ein evtl. leerer Tag erstellt
    if (((text_between->Length()) == 0) && (m_subnodes->Length() == 0))
      astring->AddUnicode(_S("/>"));
    else
      {
      i=0;
```

```
//Sonst Einfügen von '>' (als Schlusszeichen vom öffnenden Tag) zu astring
      retval = astring->AddUnichar(_S('>'));
//Solange Länge von m_subnodes > i und retvalgleich S_OK
      while (((m_subnodes->Length()) > i) && (retval==S_OK))
        {
//Kopieren von Unterknoten in m_subnodes an Position i nach abase
        retval = m_subnodes->GetAt(i, CAST(r_pointer, abase));
//Wenn o.g. Anweisung erfolgreich
        if(succeeded(retval))
          {
//abase wird mittels ToString in insert_string umgewandelt
          retval= abase->ToString(insert_string);
//Wenn o.g. Anweisung erfolgreich
          if(succeeded(retval))
            {
//Hinzufügen von insert_string an korrektem Offeset (Offset von abase + aktueller Offest) zu text_between
            retval = text_between->InsertStr(insert_string,
                     (abase->m_offset+offset));
//Offset um Länge von insert_string erhöhen
            offset+=insert_string->Length();
            }
//Löschen von abase
          SET_NULL(abase);
          }
//Inkrementieren von I, um auf nächsten Unterknoten zuzugreifen
        i++;
        }

//Hinzufügen von text_between zu astring
      astring->AddStr(text_between);
//Hinzufügen vom Schließenden Tag zu astring
      astring->AddUnichar(_S('<'));
      astring->AddUnichar(_S('/'));
      astring->AddStr(m_name);
      astring->AddUnichar(_S('>'));
      }

//Löschen von insert_string und text_between
    SET_NULL(insert_string);
    SET_NULL(text_between);

    }
  return retval;
  }


c_hresult     SC   CXMLNode::Clear                (void)
  {
//Aufrufen der Clear Funktion vom CXMLBase Objekt
  hresult retval = CXMLBase::Clear();
//Wenn o.g. Aufruf erfolgreich
  if(succeeded(retval))
    {
//Löschen von m_name
    retval = m_name->Clear();
//Wenn o.g. Anweisung erfolgreich
    if(succeeded(retval))
      {
//Löschen von m_string
      retval = m_string->Clear();
//Wenn o.g. Anweisung erfolgreich
    if(succeeded(retval))
        {
//Löschen von m_attributes
         retval = m_attributes->Clear();
//Wenn o.g. Anweisung erfolgreich
        if(succeeded(retval))
          {
//Offset auf 0 setzen
```

```
          m_offset=0;
//Wenn o.g. Anweisung erfolgreich
          if(succeeded(retval))
            {
//Löschen von m_subnodes
            retval = m_subnodes->Clear();
            }
          }
        }
      }
    }


  return retval;
  }


c_dword       SC   CXMLNode::Type                (void)
  {
//Rückgabe des Code-Wertes für ein XML-Knotenobjekt
  return XML_NODE;
  }




c_hresult     SC   CXMLNode::Get_Name            (cp_CString astring)
  {
//Setze Wert von astring auf Wert von m_name
  return astring->FromStr(m_name);
  }


c_hresult     SC   CXMLNode::Set_Name            (cp_CString astring)
  {
//Setze Wert von m_name auf Wert von astring
  return m_name->FromStr(astring);
  }



c_hresult     SC   CXMLNode::Get_Att_Val         (cp_CString aattr,
                                                  cp_CString aval)
  {
  hresult retval = S_OK;

//Wenn aval nicht leer, Lösche aval
  if (aval) retval = aval->Clear();
  else retval = E_INVALIDARG;
//Wenn o.g. Anweisung erfolgreich
  if(succeeded(retval))
    {
//Variable für Position von Attribut in m_attributes
    dword d = m_attributes->Find(aattr);
//Wenn Attribut vorhanden
    if(d != NOT_FOUND)
      {
//Variablen (Strings) für Attributname und -wert
      p_CString s1 = 0;
      p_CString s2 = 0;

//Kopieren von Attributname und –wert von m_attributes an Position d nach s1 und s2
      retval = m_attributes->GetAt(d, s1, s2);
//Wenn o.g. Anweisung erfolgreich
      if(succeeded(retval))
        {
```

```
//Löschen von s1
        SET_NULL(s1)
//Setze Wert von aval auf Wert von s2
        retval = aval->FromStr(s2);
//Löschen von s2
        SET_NULL(s2)
        }
      }
//Sonst Fehler
    else retval = E_FAIL;
    }

  return retval;
  }



c_hresult     SC   CXMLNode::Set_Att_Val          (cp_CString aattr,
                                                   cp_CString aval)
  {
  hresult retval = S_OK;
//Wenn Attribut (aattr) in m_attributes vorhanden
  if ((retval = (m_attributes->Find(aattr))) != NOT_FOUND)
    {
//Setzen von Attribut (aattr) auf übergebenen zugehörigen Wert (aval)
    retval = (m_attributes->SetAt(retval,aattr,aval));
    }
//Sonst Einfügen von Attribut (aattr und aval) in m_attributes
  else retval = m_attributes->Add(aattr,aval,0);
  return retval;
  }


c_dword       SC   CXMLNode::Get_Sub_Count        (void)
  {
//Rückgabe der Länge von m_subnodes
  return m_subnodes->Length();
  }


c_hresult     SC   CXMLNode::Get_Sub              (c_dword    aindex,
                                                   rp_CXMLBase axml)
  {
  hresult retval = S_OK;
//Wenn aindex < Länge von m_subnodes
  if (aindex < m_subnodes->Length())
    {
//Kopieren von Unterknoten von m_subnodes an Position aindex nach axml
    retval = m_subnodes->GetAt(aindex, (r_pointer)axml);
    }
//Sonst Fehler
  else retval = E_FAIL;
  return retval;
  }


c_hresult     SC   CXMLNode::Add_New_Sub_Node     (rp_CXMLNode axml)
  {
  hresult retval = S_OK;
//Hinzufügen von axml zu m_subnodes
  retval = m_subnodes->Add((r_pointer)axml);
  return retval;
  }
```

1.2.1.5.5 Kode CXML.cpp

```
//Konstruktor erstellt m_items
CXML::CXML          (c_boolean athreadsecured) : CXMLBase(athreadsecured)
  {
  m_items = new CObjectList(false);
  }

//Destruktor löscht m_items
CXML::~CXML         (void)
  {
  SET_NULL(m_items);
  }



c_hresult     SC   CXML::FromString           (cp_CString astring)
  {
//Aufrufen der FromString Funktion vom CXMLBase Objekt
  hresult retval = CXMLBase::FromString(astring);
//Variable (Zeiger auf CXMLBase Objekt) für aktuelles Objekt
  p_CXMLBase x = 0;
//Solange Länge von astring > 0
  while((retval == S_OK) && (astring->Length() > 0))
    {
//Aufruf der xml_type Funktion
//Umwandeln von astring in zugehöriges Objekt
    retval = xml_type(astring, x);
//Wenn o.g. Aufruf erfolgreich
    if(retval == S_OK)
      {
//Hinzufügen von x zu m_items
      m_items->Add(x);
      }
//Löschen von x
    SET_NULL(x);
    }
    return retval;
  }


c_hresult     SC   CXML::ToString             (cp_CString astring)
  {
//Aufrufen der ToString Funktion vom CXMLBase Objekt
  hresult retval = CXMLBase::ToString(astring);
//Wenn o.g. Aufruf erfolgreich
  if(succeeded(retval))
    {
//Variable (Zeiger auf CXMLBase Objekt) für aktuelles Objekt
    p_CXMLBase x = 0;
//Variable (Zeiger auf Stringobjekt) für aktuellen String
    p_CString s = new CString();
//Indexvariable
    dword i;
//Solange i < Länge von m_items
    for(i = 0; succeeded(retval) && (i<m_items->Length()); i++)
      {
//Kopieren von XML-Objekt von m_items an Position i nach x
      retval = m_items->GetAt(i, CAST(r_pointer, x));
//Wenn o.g. Aufruf erfolgreich
      if(succeeded(retval))
        {
//Umwandeln von x mittels ToString in s
        retval = x->ToString(s);
//Wenn o.g. Aufruf erfolgreich
        if(succeeded(retval))
          {
```

```
//Kopieren von s nach astring
          retval = astring->AddStr(s);
          }
        }
      }
//Löschen von s
    SET_NULL(s)
    }

  return retval;
  }



c_dword       SC   CXML::Type                (void)
  {
//Rückgabe des Code-Wertes für ein CXML-Objekt

  return XML_DOCUMENT;
  }


c_dword       SC   CXML::Get_Sub_Count       (void)
  {
//Rückgabe der Länge von m_items
  return m_items->Length();
  }


c_hresult     SC   CXML::Get_Sub             (c_dword    aindex,
                                              rp_CXMLBase axml)
  {
  hresult retval = S_OK;
//Wenn aindex < Länge von m_items
  if (aindex < m_items->Length())
    {
//Kopieren von XML-Objekt von m_items an Position aindex nach axml
    retval = m_items->GetAt(aindex, (r_pointer)axml);
    }
//Sonst Fehler
  else retval = E_FAIL;
  return retval;
  }


c_hresult     SC   CXML::Add_New_Sub_Node    (rp_CXMLNode axml)
  {
  hresult retval = S_OK;
//Hinzufügen von axml (Knotenobjekt) zu m_items
  retval = m_items->Add((r_pointer)axml);
  return retval;
  }


c_hresult     SC   CXML::Add_New_PI          (rp_CXMLPI axml)
  {
  hresult retval = S_OK;
//Hinzufügen von axml (Prozessanweisungsobjekt) zu m_items
  retval = m_items->Add((r_pointer)axml);
  return retval;
  }
```

1.2.2 MathClient

1.2.2.1 Modul Formular Hauptfenster

Modul MathClient-Kern Teilbeleg 2

```
'*******************************************************************************
'MathClient - Klient für MathServer
'(c) 2003 Roland Fischer, René Kreiner, Rico Roßberg, Thomas Weise, Thomas Ziegs
'Programmierer MathClient: Rico Roßberg

'Aufgrund Krankheit des Prgrammierers in der Woche vor Abgabe von TB3 mussten
'einige in TB2 geplante Eigenschaften des Programms kurzfristig geändert (gekürzt)
'werden, da wegen des Zeitdrucks Thomas Weise zur Programmierung einiger Funktionen
'einspringen musste. Wegen unterschiedlicher Programmiervorkenntnisse entwickelte
'er jedoch für die Module MathParser und XML-Verarbeitung zum Teil völlig andere
'Algorithmen als in TB2 vorgesehen...

'Die konkreten Änderungen gegenüber TB2 sind folgende:
' * Das Modul MathParser und die Funktion TreeToXMLString aus dem Modul
'   XML-Vearbeitung werden durch das Modul MathStuff ersetzt
' * Die Klasse MTreeNode und die Funktion DrawTree entfallen
' * Das Modul Kommunikation wird in den MathClient-Kern (Formular Hauptfenster)
'   integriert, da die Funktion mithilfe des ActiveX-Steuerelementes mswinsck.ocx
'   realisiert wird

'Folgende VB-Options sind in allen Modulen enthalten:
Option Base 1  'Erstes Element in Feldern hat Index 1 (statt 0)
Option Compare Text  'VB-Funktion Instr vergleicht Text ohne Groß-/Kleinschreibung
                     'zu berücksichtigen
Option Explicit  'Variablendeklaration ist erforderlich

'*******************************************************************************
'Formular Hauptfenster (Module MathClient-Kern & Kommunikation aus TB2)
'Das Hauptfenster ist das Startobjekt des Projekts und wird permanent angezeigt
'behandelt VB-üblich nur Ereignisse der enthaltenen Steuerelemente

'Im ganzen Hauptfenster gültige Variablen: das Systemkomma, Verbindungsfehler,
'Serverantwort empfangen, ausgewählter Teilausdruck, Feld aller Teilausdrücke,
'Antwort des MathServer
Dim Komma As String, ConnectionFailed As Boolean, AnswerReceived As Boolean, ATA _
    As Long, TAF As Variant, ServerAnswer As String

'*******************************************************************************
'Hauptfenster wird geladen...
'Steuerelemente werden beschriftet bzw. Vorgabewerte eingestellt
Private Sub Form_Load()
 Const C1 As String = "Konstante einfügen", C2 As String = "Operator einfügen"

 SSTab1.Tab = 0

 Me.Caption = App.Title
 Me.Command5.Caption = "Beenden"

 Me.Command1.Caption = "Formel drucken"
 Me.Command4.Caption = "Berechnen"
 'willkürlich ausgewählte Formel als Vorgabeausdruck
 Me.Text1.Text = "-(3*sin ((3/4)*Pi))+(7.3*3 log 5)^(3/4)*43.454+|3 root 7-(e^2"& _
                 ")|-(4+((7/4)*Pi))/8!"


'gewünschte Stellenzahl vom letzten Ausführen des Programms wurde gespeichert
```

```
 Me.Text2.Text = GetSettingEx("Exactness", , "100")
 Me.Command8(0).Caption = "=>"
 Me.Command8(1).Caption = "=>"

 'Combobox für Konstanten
 Me.Combo1(0).AddItem "EULERsche Zahl (e)"
 Me.Combo1(0).AddItem "LUDOLFsche Zahl (Pi)"
 'diese beiden Konstanten aus TB1 entfallen
' Me.Combo1(0).AddItem "BOLTZMANN-Konstante (k)"
' Me.Combo1(0).AddItem "Lichtgeschwindigkeit (c)"
 Me.Combo1(0).AddItem C1, 0
 Me.Combo1(0).Text = C1

 'Combobox für Operatoren
 Me.Combo1(1).AddItem "Addition (+)"
 Me.Combo1(1).AddItem "Subtraktion (-)"
 Me.Combo1(1).AddItem "Multiplikation (*)"
 Me.Combo1(1).AddItem "Division (/)"
 Me.Combo1(1).AddItem "Potenz (^)"
 Me.Combo1(1).AddItem "Wurzel (root)"
 Me.Combo1(1).AddItem "Sinus (sin)"
 Me.Combo1(1).AddItem "Kosinus (cos)"
 Me.Combo1(1).AddItem "Tangens (tan)"
 Me.Combo1(1).AddItem "Arkussinus (arcsin)"
 Me.Combo1(1).AddItem "Arkuskosinus (arccos)"
 Me.Combo1(1).AddItem "Arkustangens (arctan)"
 Me.Combo1(1).AddItem "Natürlicher Logarithmus (ln)"
 Me.Combo1(1).AddItem "Allgemeiner Logarithmus (log)"
 Me.Combo1(1).AddItem "Fakultät (!)"
 Me.Combo1(1).AddItem "Betrag (|)"
 Me.Combo1(1).AddItem C2, 0
 Me.Combo1(1).Text = C2

 Me.Text6.Text = "Ergebnis"
 Me.Command6(0).Caption = "Nächstes Teilergebnis"
 Me.Command6(1).Caption = "Vorheriges Teilergebnis"
 Me.Command2.Caption = "Aktuelles Teilergebnis drucken"
 Me.Command3.Caption = "Alle Teilergebnisse drucken"

 Me.Frame1.Caption = "Serverinformationen"
 Me.Label2.Caption = "Name:"
 Me.Label6.Caption = "Port:"
 Me.Command9.Caption = "Localhost eintragen"

 'Serverinformation wurden beim letzten Ausführen gespeichert
 Me.Text3.Text = GetSettingEx("ServerName", , Me.Winsock1.LocalHostName)
 Me.Text5.Text = GetSettingEx("ServerPort", , "20000")

 'Systemkomma ermitteln
 GetRegValue HKEY_CURRENT_USER, "Control Panel\International", "sDecimal", Komma
 Me.Label7.Caption = "Dezimaltrennzeichen:"
 Combo3.AddItem "."
 Combo3.AddItem ","
 If (Not Komma = ".") And (Not Komma = ",") Then Combo3.AddItem Komma
 Combo3.Text = Komma

 'Daten im Debug-Tab
 Text2_Change
 Text9.Text = "<?xml version=""1.0"" encoding=""iso-8859-1""?><math>"
 Text10.Text = "</math>"
 Text8.Text = ""
'willkürlich ausgewählte MathML-Daten, die nichts mit dem Ausdruck oben zu tun haben
Text7.Text = "<apply id=""0""><plus/><cn>7.4</cn><apply id=""1""><times/><cn>5" & _
             "44.9</cn><apply id=""2""><plus/><pi/><cn>4</cn></apply></apply><" & _
              "/apply>"

 Text4.Text = ""
 Text6.Text = ""
 Text12.Text = ""
```

```
 Label5.Caption = ""

End Sub

'*************************************************************************
'Komma ändern (Tab Formeleingabe)
Private Sub Combo3_Click()
 Komma = Combo3.Text
End Sub

'*************************************************************************
'Formel drucken (Tab Formeleingabe)
Private Sub Command1_Click()
 PrintText Text1.Text
End Sub

'*************************************************************************
'Daten an Server senden (Tab Debug)
Private Sub Command10_Click()
 Dim SendData As String
 On Error GoTo ErrorHandler

 ConnectionFailed = False

 'offene Verbindungen schliessen, verbinden
 If Not Winsock1.State = sckClosed Then Winsock1.Close
 Winsock1.Connect Text3.Text, Text5.Text

 'auf Verbindungsbestätigung warten
 Do While (Not (Winsock1.State = sckConnected)) And (Not ConnectionFailed)
  DoEvents
 Loop

 'Daten senden
 If Not ConnectionFailed Then
  SendData = Text11.Text & vbLf & vbLf & Text9.Text & Text7.Text & Text10.Text
  Winsock1.SendData SendData
 End If

 Exit Sub
ErrorHandler: ShowRunTimeError "Command10_Click": Resume Next

End Sub

'*************************************************************************
'Aktuelles Teilergebnis drucken (Tab Ergebnisse)
Private Sub Command2_Click()
 PrintText Text6.Text
End Sub

'*************************************************************************
'Alle Teilergebnisse drucken (Tab Ergebnisse)
Private Sub Command3_Click()
 Dim a As Long, PrintData As String

 'Alle Teilausdrücke und -ergebnisse zu einer langen Zeichenfolge zusammenfügen
 For a = 1 To UBound(TAF)
  PrintData = PrintData & "#" & a & ": " & TAF(a) & " = " & _
              FilterString(RoundedSolution(FindSpecificSolution(ServerAnswer, _
              CStr(a)), Text2.Text), ".", Komma) & vbCrLf & vbCrLf
 Next a

 PrintText PrintData

End Sub
```

```
'*****************************************************************************
'Berechnen (Tab Formeleingabe)
Private Sub Command4_Click()
 Dim XMLString As String, FString As String, a As Long, b As Long, c As Long, _
     Invalid As Boolean
 On Error GoTo ErrorHandler

 'gültige Stellenzahl überprüfen; [Invalid] wird ggf. im ErrorHandler gesetzt
 a = CLng(Text2.Text)
 If Invalid Or a < 10 Or a > 100000000 Then
  Label5.Caption = "Es muß eine Stellenzahl zwischen 10 und 100000000 angegebe" & _
                   "n werden"
  Exit Sub
 End If

 'Leerstellen rausnehmen
 Label5.Caption = "Eingegebener Ausdruck wird überprüft..."
 FString = FilterString(Text1.Text, " ")

'Folgende auskommentierte Zeilen wären normalerweise hier gekommen, wenn ich nicht
'krank geworden wäre...

' Dim MTR As MTreeNode
'
' Set MTR = MathStringToTree(FilterString(FString, Komma, "."), 1)
'
' If Not InStr(MTR.Data, "Fehler") = 0 Then
'  Label5.Caption = "Syntaxfehler an Position " & Mid(MTR.Data, 8)
'
'  Text1.SelStart = Mid(MTR.Data, 8)
'  Text1.SelLength = 1
' Else
'  XMLString = TreeToXMLString(MTR)
'  ...

 Label5.Caption = "Ausdruck wird in MathML umgewandelt..."

 'Statt dessen wird MathStuff.bas verwendet:
 Dim IDS() As String      'Feld, das die Teilausdrücke enthalten wird

 SyntaxError = False
 'jede Menge potentielle Fehlerquellen werden hier aus den Ein- und
 'Ausgabezeichenfolgen zu math2xml ausgefiltert, wie:
 ' * Systemkomma durch . ersetzen
 ' * e^x durch ex ersetzen, wie von math2xml verlangt
 ' * fehlerhafte Ausgabe eines überzähligen XML-Tags wird durch "0 + ..." ersetzt
 ' * führende Null bei 0.xxxx wird ergänzt
XMLString = FilterString(FilterString(math2xml(FilterString(FilterString(FString, _
            Komma, "."), "e^", "e"), IDS), "</>", "<plus/><cn>0</cn>"), ">.", _
            ">0.")
 If SyntaxError Then
  Label5.Caption = "Der eingegebene Ausdruck enthält einen Syntaxfehler"
  Exit Sub
 End If
 'hier ist der Unterschied zwischen geplantem MathParser und MathStuff zu Ende

 Text4.Text = FString
 Text12.Text = FString

 Text6.Text = "Die Daten werden an MathServer gesendet..."

 SSTab1.Tab = 1  'zum Tab Ergebnisse wechseln
 Label5.Caption = ""

 'der Tab Debug wird der Einfachheit halber hier zum Senden verwendet
 Text7.Text = XMLString
 Command10.Value = True
 Text6.Text = "Warte auf Antwort von MathServer..."
 TAF = IDS()
```

```
'auf Serverantwort warten (Ereignis Winsock1_DataArrival) & Zeit dafür messen
 AnswerReceived = False
 Timer1.Enabled = True

 Do While (Not AnswerReceived) And (Not ConnectionFailed)
  DoEvents
 Loop


 If AnswerReceived Then
  ATA = 1
'Gesamtergebnis anzeigen & Systemkomma wieder einsetzen
  Text6.Text = FilterString(RoundedSolution(FindSpecificSolution(ServerAnswer, _
               CStr(ATA)), Text2.Text), ".", Komma)
 ElseIf ConnectionFailed Then
  Text6.Text = "Es konnte keine Verbindung zum MathServer hergestellt werden" & _
               vbCrLf & Text8.Text
 End If

 Exit Sub
ErrorHandler: Invalid = ShowRunTimeError("Hauptfenster.Command4_Click", 13)
 Resume Next

End Sub

'*************************************************************************
'Beenden
Private Sub Command5_Click()

 'ein paar Einstellungen speichern
 SaveSettingEx "ServerName", Text3.Text
 SaveSettingEx "ServerPort", Text5.Text
 SaveSettingEx "Exactness", Text2.Text

 Unload Me

End Sub

'*************************************************************************
'Nächstes/Vorheriges Teilergebnis (Tab Ergebnisse)
Private Sub Command6_Click(Index As Integer)

 If Index = 0 Then  'Nächstes
  Increase ATA
  If ATA > UBound(TAF) Then ATA = 1
 Else   'Vorheriges
  Decrease ATA
  If ATA < 1 Then ATA = UBound(TAF)
 End If

 Text12.Text = TAF(ATA)  'Teilausdruck anzeigen
 Text6.Text = FilterString(RoundedSolution(FindSpecificSolution(ServerAnswer, _
              CStr(ATA)), Text2.Text), ".", Komma)  'Teilergebnis anzeigen

End Sub

'*************************************************************************
'Schaltflächen "=>" (Tab Formeleingabe)
Private Sub Command8_Click(Index As Integer)
 Dim a As Long, SB As String, CT As String

 CT = Combo1(Index).Text

 a = InStr(CT, "(")
 If Not a = 0 Then
 'Konstante oder Operator aus ComboBox ermitteln...
  SB = Mid(CT, a + 1, Len(CT) - a - 1)
```

```
  '..und im Eingabefeld einfügen
  If Text1.SelStart = 0 Then
   Text1.Text = Text1.Text & " " & SB & " "
  Else
   Text1.Text = Left(Text1.Text, Text1.SelStart) & " " & SB & " " & _
                Mid(Text1.Text, Text1.SelStart + 1)
  End If
 End If

End Sub

'*****************************************************************************
'Localhost eintragen (Tab Optionen)
Private Sub Command9_Click()

 Text3.Text = Me.Winsock1.LocalHostName
 Text5.Text = 20000

End Sub

'*****************************************************************************
'Änderung an der Stellenzahl in zu sendende Daten einfügen
Private Sub Text2_Change()
 Text11.Text = "GET solution decimals=""" & Text2.Text & """ fractals=""" & _
               Text2.Text & """ HTTP/1.1"
End Sub

'*****************************************************************************
'hier wird die Zeit bis zum Eintreffen der Serverantwort abgeschätzt, auf Basis
'vorher gemessener Zeiten, der Anzahl der Operatoren im gesendeten Ausdruck und der
'gewünschten Stellenzahl
'wunderbar genau für den unter Form_Load vorgebenen Ausdruck, aber sonst...
Private Sub Timer1_Timer()
 Dim T As Single, SI As Single, TR As Long, RT As String
 Static Initialized As Boolean, StartTimer As Single, ElapsedTime As Single, _
        CalcSpeed As Double

 If (Not AnswerReceived) And (Not ConnectionFailed) Then     'warten und anzeigen
  T = Timer

  If Not Initialized Then
   CalcSpeed = CDbl(GetSettingEx("CalcSpeed", , "0"))
   StartTimer = T
   Initialized = True
  End If

  If T < StartTimer Then Increase T, 86400
  ElapsedTime = T - StartTimer

  RT = RoundedTime(CDbl(CalcSpeed * (UBound(TAF) * CLng(Text2.Text) ^ 3) - _
                   ElapsedTime), 2)
  If Not RT = "" Then Text6.Text = "Warte auf Antwort von MathServer..." & _
                                   vbCrLf & "Noch ca. " & RT
 Else    'fertig und Mittelwert schätzen
  Initialized = False
  Timer1.Enabled = False

  If Not UBound(TAF) = 0 Then SI = ElapsedTime / (UBound(TAF) * _
                                   CLng(Text2.Text) ^ 3)

  TR = CLng(GetSettingEx("TimesRun", , "0"))
  CalcSpeed = (TR * CalcSpeed + SI) / (TR + 1)
  Increase TR

  SaveSettingEx "CalcSpeed", CalcSpeed
  SaveSettingEx "TimesRun", TR
 End If

End Sub
```

```
'*******************************************************************************
'Empfang der Serverantwort, in Blöcken entsprechend der Systemeinstellung
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
 Dim RecData As String
 Static AllData As String

 Winsock1.GetData RecData    'Block empfangen
 AllData = AllData & RecData     'zusammenfügen

'Empfang abgeschlossen (gültige Endemarkierung in Antwort) ?
 If (Not InStr(RecData, "</math>") = 0) Or (Not InStr(RecData, "ERROR") = 0) Then
  ServerAnswer = AllData
  Text8.Text = AllData  'Achtung: Textboxen fassen nur 32 kB !
  Winsock1.Close     'trennen
  AllData = ""
  AnswerReceived = True 'wartende DoEvents-Schleifen anderswo freigeben
 End If

End Sub

'*******************************************************************************
'Verbindungsfehler
Private Sub Winsock1_Error(ByVal Number As Integer, Description As String, ByVal _
            Scode As Long, ByVal Source As String, ByVal HelpFile As String, _
            ByVal HelpContext As Long, CancelDisplay As Boolean)

 Text8.Text = "WinSock-Fehler " & Number & ": " & Description
 ConnectionFailed = True

End Sub
```

1.2.2.2 Modul Grundlagen

```
'*******************************************************************************
'Modul Grundlagen
'enthält außer den folgenden Funktionen noch weitere Hilfsfunktionen, die aber hier
'nicht vollständig wiedergegeben werden, da sie nur allgemeine, nicht aber auf den
'MathClient spezialisierte Funktionalität besitzen (ähnlich Modul Basics im
'MathServer)

'Schnittstelle und kurze Beschreibung dieser Hilfsfunktionen:
' * Sub Decrease(Number As Variant, Optional DecStep As Variant = 1)
'   Erniedrigt [Number] um [DecStep]
' * Function FilterString(InText As String, RemoveStr As String, Optional _
'         ReplaceStr As String) As String
'   Entfernt [RemoveStr] aus einer Zeichenfolge, oder ersetzt es durch [ReplaceStr]
' * Function GetSettingEx(Key As String, Optional Section As String, Optional _
'         Default As String, Optional HKLM As Boolean) As Variant
'   Liest einen Wert aus dem Schlüssel der Anwendung in der Registrierung
' * Sub Increase(Number As Variant, Optional IncStep As Variant = 1)
'   Erhöht [Number] um [IncStep]
' * Function RoundedTime(Seconds As Double, Optional MaxLength As Long, Optional _
'         Exactness As Long) As String
'   Rechnet [Seconds] in andere Zeiteinheiten um
' * Sub SaveSettingEx(Key As String, Setting As Variant, Optional Section As _
'      String, Optional HKLM As Boolean)
'   Speichert einen Wert im Schlüssel der Anwendung in der Registrierung
' * Function ShowRunTimeError(ErrorLocation As String, Optional Exception) As _
'         Boolean
'   Zeigt Laufzeitfehler an, außer durch [Exception] ausgeschlossene

'*******************************************************************************
'Druckt eine Zeichenfolge auf dem Standarddrucker aus

Sub PrintText(PrintData As String)
 Dim SD As Printer, a As Long, PD As String, CPD As String, RE As Boolean, b _
     As Long, c As Long, d As Long, CPD2 As String, e As Long

 PD = PrintData
 Set SD = Printer    'Standarddrucker auswählen

 'folgende Schleife fügt nur zusätzliche Zeilenumbrüche ein...
 b = 1
 Do
  a = InStr(b, PD, vbCrLf)
  If a = 0 Then
   CPD = Mid(PD, b)
   a = Len(CPD)
  Else
   CPD = Mid(PD, b, a - b)
  End If

  c = SD.TextWidth(CPD)

  If c > SD.ScaleWidth Then
   CPD2 = CPD
   e = SD.ScaleWidth / c * Len(CPD) - 5

   For d = 1 To Int(c / SD.ScaleWidth)
    CPD = Left(CPD, d * e + (d - 1) * 2) & vbCrLf & Mid(CPD, d * e + (d - 1) _
          * 2 + 1)
   Next d

   PD = FilterString(PD, CPD2, CPD)
   Increase a, Len(CPD) - Len(CPD2)
  End If

  b = a + 1
```

```
  RE = (b >= Len(PD))
 Loop Until RE

 SD.Print PD      'zu druckende Daten
 SD.EndDoc   'zum Drucken freigegeben

End Sub

'*****************************************************************************
'Rundet ein Teilergebnis auf die gewünschte Stellenzahl
Function RoundedSolution(FullLengthSolution As String, MaxExactness As Long) _
         As String
 Dim OutString As String, KP As Long, MSE As Long, Op As Long, CO As Boolean

 OutString = FullLengthSolution
 MSE = MaxExactness

 KP = InStr(OutString, ".") 'Kommaposition & Komma rausnehmen
 If Not KP = 0 Then OutString = FilterString(OutString, ".")

 If Len(OutString) > MSE Then    'abrunden (abschneiden)...
  If CLng(Mid(OutString, MSE + 1, 1)) < 5 Then
   OutString = Left(OutString, MSE)
  Else  '... oder kompliziertes Aufrunden, ggf. bis hin zur ersten Stelle
   OutString = Left(OutString, MSE)
   Op = Right(OutString, 1)
   Increase Op

   If Op > 9 Then
    CO = True
    OutString = "0" & OutString
    Increase KP

    Do While CO
     Op = Right(OutString, 1)
     Increase Op
     CO = (Op > 9)
     If CO Then
      OutString = Left(OutString, Len(OutString) - 1)
     Else
      OutString = Left(OutString, Len(OutString) - 1) & CStr(Op)
     End If
    Loop

    If Left(OutString, 1) = "0" Then
     OutString = Mid(OutString, 2)
     Decrease KP
    End If

    If KP > Len(OutString) + 1 Then
     OutString = OutString & String(KP - Len(OutString) - 1, "0")
    End If
   Else
    OutString = Left(OutString, MSE - 1) & CStr(Op)
   End If
  End If
 End If

 'Komma wieder einfügen
 If (Not KP = 0) And (Not KP > Len(OutString)) Then OutString = Left(OutString, _
    KP - 1) & "." & Mid(OutString, KP)

 RoundedSolution = OutString

End Function
```

1.2.2.3 Modul XML-Verarbeitung

```
'*****************************************************************************
'Rest vom Modul XML-Vearbeitung

'*****************************************************************************
'Findet das Teilergebnis mit der ID [ID]
Function FindSpecificSolution(SearchString As String, ID As String) As String
 Dim a As Long, b As Long, c As Long, IDS As String, OutString As String

 IDS = "<cn id=""" & ID & """>" 'cn auf
 c = Len(IDS)

 a = InStr(SearchString, IDS)
 If a = 0 Then
  OutString = "Fehler: Eingabefehler oder nicht definierte Lösung"
 Else
  b = InStr(a, SearchString, "</cn>")    'cn zu
  If b = 0 Then
   OutString = "Fehler: Fehlerhafte Serverantwort"
  Else
   OutString = Mid(SearchString, a + c, b - a - c)
   If OutString = "Fehler" Then OutString = _
      "Fehler: Eingabefehler oder nicht definierte Lösung"
  End If
 End If

 FindSpecificSolution = OutString

End Function
```

1.2.2.4 Modul MathStuff

Aus bereits erwähnter schwerer Erkrankung von Rico Roßberg fiel dem Teamleiter die Aufgabe zu, primäres Funktionieren des Klienten zu ermöglichen. Die Umwandlung der Infix-Eingabe in MathML war die einzige noch zu ergänzende Implementation. Da der Teamleiter sich leider nicht in bereits konkrete und fortgeschrittene Planung von Herrn Roßberg hineinfinden konnte, wurde behelfsmäßig das Modul MathStuff erstellt. Es wird hier nicht tiefergehend kommentiert, da es das Produkt einer einzigen Nacht ist und der Teamleiter andere Partikulärtätigken, seine nativen Module tangierend, priorisiert bearbeiten muss.

```
Attribute VB_Name = "MathStuff"
'Option Base 1 'Erstes Element in Feldern hat Index 1 (statt 0)
Option Compare Text 'Funktion "Instr" vergleicht Text ohne Groß-/Kleinschreibung zu berücksichtigen
Option Explicit 'Variablendeklaration ist erforderlich

'einige Konstanten benötigt zur Verarbeitung
Private Const num = "?0123456789.", allowed = "abcdefghijklmnopqrstuvwxyz" + num

Public SyntaxError As Boolean


'Prüft ob ein angegebener Ausdruck eine gültige Zahl darstellt
Function isnumber(ByVal S As String) As Boolean
Dim b As Boolean
Dim I As Long
b = False
I = 1
If (Left(S, 1) = "+") Or (Left(S, 1) = "-") Then Increase I
For I = I To Len(S)
  If InStr(num, Mid(S, I, 1)) = 0 Then GoTo ende
Next
b = True
ende:
isnumber = b
End Function


'parst einen xmath-Ausdruck (siehe unten) zu einer Formel zurück, damit
'diese dann im Teilergebnisse Fenster dargestellt werden kann
Function reparse(ByVal S As String) As String

Dim I As Long
Dim x As String
Dim j As Long
Dim k As Long
Dim p As Long

While Left(S, 1) = "("
 S = Mid(S, 2, Len(S) - 2)
 Wend

I = InStr(S, "(")
If I > 0 Then
 x = Left(S, I - 1)
 S = Mid(S, I)

 While Left(S, 1) = "(("
 S = Mid(S, 2, Len(S) - 2)
 Wend

 k = 0
 p = 0
 j = 1
 Do
  If (Mid(S, j, 1) = "(") Or (Mid(S, j, 1) = "[") Then Increase k
```

```
  If (Mid(S, j, 1) = ")") Or (Mid(S, j, 1) = "]") Then Decrease k
  If (k <= 1) And (Mid(S, j, 1) = ";") Then p = j
  Increase j
 Loop While (j < Len(S)) And (k <> 0)

 S = Mid(S, 2, j - 2)

 If x = "abs" Then
 x = "|" + reparse(S) + "|"
 Else
 If x = "factorial" Then
 x = reparse(S) & "!"
 Else

 If x = "plus" Then
 x = reparse(Mid(S, 1, p - 2)) & "+" & reparse(Mid(S, p))
 Else
 If x = "minus" Then
 x = reparse(Mid(S, 1, p - 2)) & "-" & reparse(Mid(S, p))
 Else
 If x = "times" Then
 x = reparse(Mid(S, 1, p - 2)) & "*" & reparse(Mid(S, p))
 Else
 If x = "divide" Then
 x = reparse(Mid(S, 1, p - 2)) & "/" & reparse(Mid(S, p))
 Else
 If x = "root" Then
 x = reparse(Mid(S, 1, p - 2)) & "root" & reparse(Mid(S, p))
 Else
 If x = "log" Then
 x = reparse(Mid(S, 1, p - 2)) & "log" & reparse(Mid(S, p))
 Else
 If x = "power" Then
 x = reparse(Mid(S, 1, p - 2)) & "^" & reparse(Mid(S, p))
 Else
 x = x & "(" & reparse(S) & ")"
 End If
 End If
 End If
 End If
 End If
 End If
 End If
 End If
 End If
 Else
 x = S
 End If

 reparse = x
End Function
```

'fügt einen neuen Unterausdruck an den Array ids an, dabei ist der Index des Unterausdrucks stets 'auch seine ID im MathML-Text

```
Public Function addid(ByRef IDS() As String, S As String) As Long
ReDim Preserve IDS(UBound(IDS) + 1)
IDS(UBound(IDS)) = S
addid = UBound(IDS)
End Function
```

```
'wandelt einen xmath-Ausdruck in einen MathML (xml) Text um
'xmath ist ein völlig fiktives Zwischenprodukt des Parsevorgangs der z.b. führt
'3 * 5 + 7 log 3 zu add(mul(3;5);log(7;3))
'was dann viel leichter in XML zerlegt werden kann
'wir gehen also vom Infix erst zum vollständig geklammerten dyadischen Ausdruck
'und dann zum MathML

Public Function xmath2xml(ByVal S As String, ByRef IDS() As String) As String

Dim x As String
Dim Y As String
Dim ID As Long
x = ""
Y = ""

While Left(S, 1) = "("
S = Mid(S, 2, Len(S) - 2)
Wend

If S = "pi" Then
x = "<pi/>"
Else
If isnumber(S) Then
x = "<cn>" & S & "</cn>"
Else

If (Left(S, 1) = "-") Then
 S = Right(S, Len(S) - 1)
 ID = addid(IDS, reparse("minus(0;" & S & ")"))
 x = "<apply id=""" & ID & """><minus/><cn>0</cn>"
 Y = "</apply>"
 End If

ID = addid(IDS, reparse(S))
x = x & "<apply id=""" & ID & """>"

Dim I As Long
Dim j As Long
Dim k As Long
I = InStr(S, "(")

x = x & "<" & Left(S, I - 1) & "/>"
S = Mid(S, I + 1)

k = 0
I = 1
While ((k > 0) Or (Mid(S, I, 1) <> ";")) And (I < Len(S))
If Mid(S, I, 1) = "(" Then Increase k
If Mid(S, I, 1) = ")" Then Decrease k
Increase I
Wend

x = x & xmath2xml(Left(S, I - 1), IDS)
S = Mid(S, I + 1)
If S <> "" Then
S = Left(S, Len(S) - 1)
x = x & xmath2xml(S, IDS)
End If
x = x & "</apply>"
End If
End If
xmath2xml = x + Y
End Function
```

```
'es wird nach links und rechts sondiert, um das Ende eines Operanden im Infix auf-
'zuspüren

Function leftsond(ByVal S As String, ByVal I As Long)
Dim k As Long
If I <= 0 Then
 I = 0
 GoTo ok
 End If
k = 0
Do
If (Mid(S, I, 1) = "(") Or (Mid(S, I, 1) = "[") Then Increase k
If (Mid(S, I, 1) = ")") Or (Mid(S, I, 1) = "]") Then Decrease k
Decrease I
If I = 0 Then GoTo ok
Loop While (I > 0) And ((k <> 0) Or (InStr(allowed, Mid(S, I, 1)) <> 0))
ok:
leftsond = I
End Function


Function rightsond(ByVal S As String, ByVal I As Long)
Dim k As Long
k = 0
If Mid(S, I, 1) = ";" Then GoTo ok
Do
If (Mid(S, I, 1) = "(") Or (Mid(S, I, 1) = "[") Then Increase k
If (Mid(S, I, 1) = ")") Or (Mid(S, I, 1) = "]") Then Decrease k
Increase I
Loop While (I < Len(S)) And ((k > 0) Or (InStr(allowed & "[(", Mid(S, I, 1)) <> 0))
ok:
rightsond = I
End Function

'explode ersetzt im String S einen Operator an Stelle i durch den Ausdruck plc,
'wobei nach links (L) und/oder rechts (R) nach Operanden sondiert werden kann
'dadurch werden binäre und rechts- sowie lings-unäre Ausdrücke mit einem
'gemeinsamen Algorithmus abhandelbar (s wird dadurch größer ⇒ explodiert)
Function explode(ByVal S As String, ByVal plc As String, ByVal I As Long, _
                 ByVal L As Boolean, ByVal R As Boolean) As String
Dim j As Long
Dim k As Long
Dim x As String
Dim Y As String

If L Then
 j = leftsond(S, I - 1)
Else
 j = I - 1
 End If

 If R Then
 If InStr(allowed & ";", Mid(S, I, 1)) <> 0 Then
  k = rightsond(S, I + 1)
Else
k = rightsond(S, I)
End If
Else
k = I
End If

If j > 0 Then
x = Left(S, j)
Else
x = ""
End If
```

```
If k < Len(S) Then
Y = Right(S, Len(S) - k + 1)
Else
Y = ""
k = k + 1
End If

S = x & "[" & plc & "(" & Mid(S, j + 1, k - j - 1) & ")]" & Y
explode = S
End Function


'explode-replace, wandelt alle Operatoren der Art src mit explode um

Function exploderepl(ByVal S As String, ByVal src As String, ByVal rpl As String, _
                ByVal L As Boolean, ByVal R As Boolean) As String
Dim I As Long
Dim x As String
I = InStr(S, src)
While I > 0
If L And R Then x = ";"
S = explode(Left(S, I - 1) & x & Mid(S, I + Len(src)), rpl, I, L, R)
I = InStr(S, src)
Wend
exploderepl = S
End Function


'do-abs, verarbeitet die Betragsklammern |

Function dabs(ByVal S As String) As String
Dim I As Long
Dim j As Long
Dim k As Long
I = InStr(S, "|")
While I > 0
 j = I + 1
 Do
 If (Mid(S, j, 1) = "(") Or (Mid(S, j, 1) = "[") Then Increase k
 If (Mid(S, j, 1) = ")") Or (Mid(S, j, 1) = "]") Then Decrease k
 Increase j
 Loop While (j < Len(S)) And ((k <> 0) Or (Mid(S, j, 1) <> "|"))

S = Left(S, I - 1) & "[abs(" & Mid(S, I + 1, j - I - 1) & ")]" & Mid(S, j + 1)
I = InStr(S, "|")
Wend
dabs = S
End Function


'plus-minus-john-do, unterscheidet unäre + und – von binären, wobei die unären
'Pluse gleich gelöscht und die Minuse in ‚?' umgewandelt werden, damit sie
'später keinen Schaden mehr anrichten.

Function pmjohndo(ByVal S As String)

Dim I As Long

For I = 1 To Len(S)
If (Mid(S, I, 1) = "-") Or (Mid(S, I, 1) = "+") Then
If (I = 1) Then GoTo ok
If (((InStr(num, Mid(S, I + 1, 1)) <> 0) And (InStr(num & ")]", Mid(S, I - 1, 1)) =
0))) Then
ok:
If (Mid(S, I, 1) = "+") Then
S = Left(S, I - 1) + Mid(S, I + 1)
Else
S = Left(S, I - 1) + "?" + Mid(S, I + 1)
End If End IfEnd If Next
```

```
pmjohndo = S
End Function


'math2xmath wandelt einen Infix-Ausdruck in das völlig fiktive Zwischenprodukt xmath um
'dabei werden zuerst die Betragsklammern abgehandelt, dann die Operatoren
'ihrer Wertigkeit nach. Um Fehler in der exploderepl-Routine durch mehrfaches
'Auftreten von Operatoren wie sin in arcsin und e in Power zu verhindern,
'werden diese zwischenzeitlich in lustige Ausdrücke umgewandelt
'um falsches Klammern zu verhinden, werden zusätzliche Klammerungen durchgeführt

Function math2xmath(ByVal S As String)

S = dabs(S)

S = exploderepl(S, "^", "pow#r", True, True)
S = exploderepl(S, "e", "#xp", False, True)

S = exploderepl(S, "/", "divid#", True, True)
S = exploderepl(S, "*", "tim#s", True, True)

S = exploderepl(S, "arcsin", "arcxin", False, True)
S = exploderepl(S, "sin", "xin", False, True)
S = exploderepl(S, "arccos", "arcxos", False, True)
S = exploderepl(S, "cos", "xos", False, True)
S = exploderepl(S, "arctan", "arcxan", False, True)
S = exploderepl(S, "tan", "xan", False, True)

S = exploderepl(S, "ln", "xn", False, True)
S = exploderepl(S, "log", "xog", True, True)
S = exploderepl(S, "root", "xoot", True, True)

S = exploderepl(S, "!", "factorial", True, False)

S = pmjohndo(S)

S = exploderepl(S, "+", "plus", True, True)
S = exploderepl(S, "-", "minus", True, True)

S = FilterString(S, "?", "-")
S = FilterString(S, "xan", "tan")
S = FilterString(S, "#", "e")
S = FilterString(S, "xoot", "root")
S = FilterString(S, "xn", "ln")
S = FilterString(S, "xog", "log")
S = FilterString(S, "xan", "tan")
S = FilterString(S, "xos", "cos")
S = FilterString(S, "xin", "sin")
S = FilterString(S, "[")
S = FilterString(S, "]")

math2xmath = S
End Function

'Die Routine führt die Umwandlung des Infixausdrucks über die xmath-Zwischenstufe
'nach XML aus
Function math2xml(ByVal S As String, ByRef IDS() As String) As String

On Error GoTo ErrorHandler

Dim I As Long
I = 1
ReDim IDS(0)
math2xml = xmath2xml(math2xmath(S), IDS)

Exit Function
ErrorHandler: SyntaxError = ShowRunTimeError("xmath2xml", 5): Resume Next
End Function
```

1.3 Testplan

Zuerst wird der Server einzeln getestet. Mit Hilfe von Pythonscript werden die Testdaten (Rechenaufgaben) zum Server geschickt. Die Matheaufgaben werden wie spezifiziert in MathML Syntax zu übertragen. So werden zuerst alle implementierten Mathefunktionen einzeln getestet. Danach wird dazu übergegangen mehrere Funktionen in Kombination zu testen, welche Funktionen zusammen getestet werden geschieht nachdem Zufallsprinzip. Die Überprüfung der Ergebnisse erfolgt mittels Taschenrechner. Der letzte Test zielt auf die Multithreadfunktionalität ab. Z.B. Pyhton wird 3x gestartet und jeweils ein Rechenscript geöffnet, die Berechnung von Sin 0,56 bei 500 Vorkommastellen und 500 Nachkommastellen damit die Berechnung lang genug dauert. Die 3 Scripte werden so schnell wie möglich hintereinander ausgeführt.
Erfolgen diese Tests erfolgreich, erfolgt gleiche Vorgehensweise bei dem Klient.

Unter dem Punkt 1.4 Systemtest ist eine Tabelle aufgeführt, welche nur einige repräsentative Ergebnisse enthält.

Die Tests führte Thomas Ziegs durch, Test dauerten ca. 2 Tage.
Die Ergebnisse wie z.B. 1,9999999999…. welche der Server liefert, die eigentlich eine 2 darstellen kommen durch die irrationalen bzw. periodischen Binärzahlen zustande.

1.3.1 Testfälle

Die Testfälle MathServer und MathClient werden kombiniert betrachtet und gemeinsam dargestellt. Da die Applikation nur einer Aufgabe, der Berechnung von Ausdrücken, dient, gibt es im Prinzip nur diese beiden.
Mit der Testung der einzelnen Module Klient und Server wird gleichzeitig die Richtigkeit der Gesamtapplikation untermauert.
Wir testen sowohl Klient und Server welche gleichzeitig auf dem selben Rechner laufen und über eine TCT/IP-Verbindung (127.0.0.1) auf Localhost verbinden als auch den Betrieb im Netzwerk (Windows XP, Windows 2000) und erhalten exakt die gleichen Ergebnisse unter beiden Betriebssystemen. Die Verzögerungszeit durch das Netzwerk (10 MBit Ethernet LAN) stellt sich als vernachlässigbar heraus. Daher soll nachfolgend weder Netzwerk-/Lokalbetrieb noch Betriebssystem unterschieden werden.

1.3.3 Zeitplan und Verantwortlichkeiten

Der Verantwortliche für die Tests war Thomas Ziegs, ihm oblag die zeitliche Planung und Durchführung der Tests. Zuerst werden die Tests des Servers separat, dann in Kombination mit dem Klienten durchgeführt.

1.3.4 Beispieltestskript in Python

Mit solchen Skripts kann der Server separat getestet warden.

```
#with this python script
#the server can be tested
#server should run at port 20000

from socket import *

s = socket(AF_INET, SOCK_STREAM)
s.connect(("127.0.0.1",20000))

formel = "GET solution decimals=\"50\" fractals=\"500\" HTTP/1.0\n\n"
formel = formel + "<?xml version=\"1.0\" encoding=\"UTF-8\"?>"
formel = formel + "<math><apply id=\"1\">"
formel = formel + "<plus/>"
formel = formel + "<cn>0</cn><pi/>"

formel = formel + "</apply></math>"

print formel

s.send(formel)

answer = s.recv(999999)
print answer
```

1.4 Systemtest

Hier werden die Testresultate des Servers alleine, des Servers angesteuert mit dem mitgelieferten Klienten und die eines handelsüblichen Taschenrechners gegenübergestellt.

Tabelle 1.4.1

Funktion	Parameter	Ergebnis Server	Ergebnis Klient	Ergebnis TR
abs	5 -5	5 5	5 5	5 5
arccos	0,33 -0,43	1,2344927516 2,0152891037	1,2344927516 2,0152891037	1,234492751 2,015289104
arcsin	0,2 -0,66	0,2013579207 -0,720818760	0,2013579207 -0,7208187608	0,201357921 -0,72081876
arctan	0,3 -0,87	0,2914567944 -0,7159911144	0,2914567944 -0,7159911144	0,2914567945 -0,7159911144
cos	0,2 -4,47 1000	0,9800665778 -0,2400224532 0,5623790762	0,9800665778 -0,2400224532 0,5623790762	0,9800665778 -0,2400224533 0,5623790763
divide	5 ; 2 10 ; 0	2,5 Fehler	2,5 Fehler	2,5 ERROR
exp	0 5	1 148,4131591025	1 148,4131591025	1 148,4131591
factorial	5 -5	120 Fehler	120 Fehler	120 ERROR
ln	-1 0 2	Fehler Fehler 0,6931471805	Fehler Fehler 0,6931471805	ERROR ERROR 0,6931471806
log	10 ; -1 10 ; 0 10 ; 2 2 ; 2	Fehler Fehler 0,3010299956 1	Fehler Fehler 0,3010299956 1	ERROR ERROR 0,3010299957 1
minus	5 ; 5 4 ; -6 0 ; 3	0 10 -3	0 10 -3	0 10 -3
pi	-	3,1415926535	3,1415926535	3,141592654
plus	5 ; 5 1000 ; -10000 0 ; -9,2	10 -9000 -9,1999999999	10 -9000 -9,2	10 -9000 -9,2
power	0 ; 0 0 ; 1 5 ; 2	Fehler 0 24,9999999999	Fehler 0 24,9999999999	ERROR 0 25
quotient	6 ; 3 5,2 ; 0 5,2 ; 3	2 ; 0 Fehler 1 ; 2,2	- - -	- - -
root	-2 ; 5 0 ; 5 1 ; 5 2 ; -5 2 ; 9 2,5 ; 9	0,4472135954 Fehler 4,9999999999 Fehler 2,9999999999 2,4082246852	0,4472135954 Fehler 5 Fehler 3 2,4082246852	0,4472135955 ERROR 5 ERROR 3 2,408224685

sin	0 2 -2 1000	0 0,9092974268 -0,9092974268 0,82687954053	0 0,9092974268 -0,9092974268 0,82687954053	0 0,9092974268 -0,9092974268 0,8268795405
tan	0 1 -1,7 1000	0 1,5574077246 7,6966021394 1,4703241557	0 1,5574077246 7,6966021394 1,4703241557	0 1,557407725 7,696602139 1,470324156
times	5 ; 5 0,2 ; -6 1000 ; 0,0003 5000 ; 1	25 -1,1999999999 0,2999999999 5000	25 -1,2 0,3 5000	25 -1,2 0,3 5000
$\log(\sin(12/7+1{,}35))$	-	-1,1122141905	-1,1122141905	-1,112214191
$e^{\arctan(\mathrm{abs}(-3))}$	-	3,4870139644	3,4870139644	3,487013964
$\ln(3{,}22*7+10)$	-	3,4824701017	3,4824701017	3,482470102
$\sqrt{\log(10^4)}$	-	1,9999999999	1,9999999999	2

1.5 Abschlusseinschätzung

Nach Beendigung der Systemtests stellt sich eine vollständige Funktionalität des Projekts gemäß Spezifikation Teilbeleg 1 dar.

Auf Grund der plötzlichen, schwerwiegenden Erkrankung des Teammitglieds Rico Roßberg wurden einige Funktionen des Klienten spartanischer realisiert als anfänglich erhofft. Sie behalten zwar ihre volle, teilbeleggemäße Funktionalität, fallen jedoch etwas gröber aus.
Gegenüber Teilbeleg 2 mussten beim Klienten in der Strukturierung auf Grund obiger Erkrankung größere Abweichungen hingenommen werden, da der Teamleiter kurzzeitig einspringen musste. (siehe Module).

Beim Server wurden nur minimale Abstriche gemacht. Die Fehlerbehandlung ist z.B. noch zu erweitern. Die Hauptfunktion des Programms, die Berechnung auf beliebiger Breite mit voller Unicode-, XML-, MathML-, HTTP-, TCP/IP- und ANSI-Kompatibilität der Datenströme konnte jedoch voll erfüllt werden. Auch wurde das Kriterium der Rechenzeit von bis zu einer Minute für Rechnungen hoher Genauigkeit (≤ 1000 Stellen) durch hohe Optimierung erfüllt. Es ist möglich, mit beliebig vielen Klienten (sofern das die physikalischen Parameter des Servers zulassen) auf den Server verbinden. Dazu ist die minimale Größe des Servers und aller seiner Module (< 300 Kilobytes) zu beachten sowie eine enorme Wiederverwert- und Erweiterbarkeit des Quellkodes für spätere Versionen.

Alle Anforderungen wurden in vollem Umfang erfüllt.

2 Systemhandbuch

Das Systemhandbuch wurde als Anhang separat mitgeliefert.

3 Übersicht über die Arbeitsaufgaben aller Teammitglieder bei der Projektbearbeitung

Roland Fischer

Roland Fischer stellte der XML-Arbeitsgruppe seine algorithmentechnischen Fähigkeiten zur Verfügung. Unter seiner Mitwirkung entstanden die Eingangs- und Ausgangsmodule des Projekts, die XML-Generation und –Zerlegung.
Er war zudem verantwortlich für die Zeitplanung und Systemanalyse im Rahmen des ersten Teilbelegs sowie das Benutzerhandbuch.
Ebenso wirkte er an der Installationssoftware mit.

René Kreiner

Mit seiner Erfahrung in der C-Programmierung brachte René Kreiner das nötige Know-How zur Umsetzung des XML-Parsers in das Team ein und entwickelte maßgeblich dessen Klassenhierarchie und das Konzept der gegenseitigen Rekursion des xml_type-Gefüges.
Er war federführend bei der textuellen Umsetzung der Teilbelegerzeugnisse der XML-Arbeitsgruppe.

Rico Roßberg

Rico Roßberg bereicherte das Team durch großen Sachverstand in der grafischen Programmierung sowie durch die Entwicklung des Formelparsers, welche überplanmäßig schnell erfolgte.
Ähnliches Tempo legte er bei Erstellung der Klientenapplikation an den Tag, bis er unerwartet schwer (ärztlich nachweisbar) erkrankte. Dennoch beendete er die Arbeit am Klienten erfolgreich.

Thomas Weise

Thomas Weise übernahm die Tätigkeit als Teamleiter, Hauptarchitekt und Entwurfsverantwortlicher. Seine Grundobjekte und die Engine ermöglichten die gute modulare Zusammenarbeit des Teams.

Thomas Ziegs

Thomas Ziegs entwickelte mit Hilfe der Assemblersprache und unter Aufwendung umfangreicher Nachforschungen die mathematischen Routinen des Projektes.
Er führte die Systemtests durch, wobei er die letzten, kleinen Ecken und Kanten des Projektes mit äußerster Akribie abschliff.
Danach unterstützte er das Installationsteam mit seiner Fähigkeit auf dem Gebiet der NSIS-Skripts.

Unter Anwendung vieler wichtigen neuzeitlichen Programmiertechniken wie z.B. Assembler-Optimierung, Dynamische Link Bibliotheken, Objektorientierte Programmierung mit direkter Manipulation der Vererbungsmechanismen, Referenzzählung, Netzwerkprogrammierung, Multithreading, Integration verschiedener Programmiersprachen (Phyton, C++, Visual Basic), teamorientiertes Arbeiten auf einer netzwerkgestützten Hauptkodebasis, Direktzugriff auf das Betriebssystem, selbstständiges Implementieren weltweit anerkannter Standards und der Entwicklung eines Client-Server-Systems wurde ein Projekt geschaffen, dessen Wertigkeit den Anforderungen des Kunden, der Gruppendynamik sowie dem durchaus wissenschaftlichen Rahmen des Softwarepraktikums genüge tut.

Roland Fischer, René Kreiner, Rico Roßberg, Thomas Weise, Thomas Ziegs

SWP11

2003

Handbuch

MathServer & Client

Version 1

2003

Inhaltsverzeichnis

1. Vorwort 3

2. Systemanforderungen 3

   2.1 MathServer (Mindestanforderung) 3

   2.2 Mathserver (Empfohlen) 3

   2.3 MathClient (Mindestanforderung) 3

3. Installation 4

4. MathServer 6

   4.1 Bedienung 6

   4.2 Unterstütze Funktionen 6

5. MathClient 7

   5.1 Bedienung 7

   5.2 Unterstütze Funktionen 10

6. Deinstallation 11

1. Vorwort

Dieses Programm entstand während eines Softwarepraktikums an der Technischen Universität Chemnitz unter der sachkundigen Führung des Dipl. Inf. Neugebauer.

Wir unterstützen viele weltweit anerkannte Standards wie MathML, XML, HTTP und TCP/IP um eine allgemeingültige Plattform für Berechnungen zu bieten.

Wir von MathServer-Development haben uns zum Ziel gesetzt, unseren Nutzern das bestmögliche Rechenerlebnis zu bieten.

2. Systemvoraussetzungen

Als Betriebssystem wird sowohl für den Client als auch für den Server Windows NT 4.0, Windows 2000 oder Windows XP vorausgesetzt. Als sonstige Basissoftware ist Visual Basic Runtime Libraries für den Client notwendig. Diese werden mit dem Installer mitgeliefert.

2.1 MathServer (Mindestanforderung)

• 400 MHz CPU
• 64 MB RAM
• 30 MB freier Festplattenspeicher
• CD-ROM Laufwerk
• Netzwerkzugang (LAN), min. 10 MBit/s bzw. Internetzugang, Modem 56 KBit/s

2.2 Mathserver (Empfohlen)

• 1 GHz CPU
• 256 MB RAM
• 100 MB freier Festplattenspeicher
• CD-ROM Laufwerk
• Netzwerkzugang (LAN), 100 MBit/s bzw. Internetzugang, DSL

2.3 MathClient (Mindestanforderungen)

• 400 MHz CPU
• 64 MB RAM
• 10 MB freier Festplattenspeicher
• CD-ROM Laufwerk
• Netzwerkzugang (LAN), min. 10 MBit/s bzw. Internetzugang, Modem, 56 KBit/s
• Monitor 15.., 70 Hz und Grafikkarte, 256 Farben bei einer Auflösung von 800x600
• Tastatur, Maus

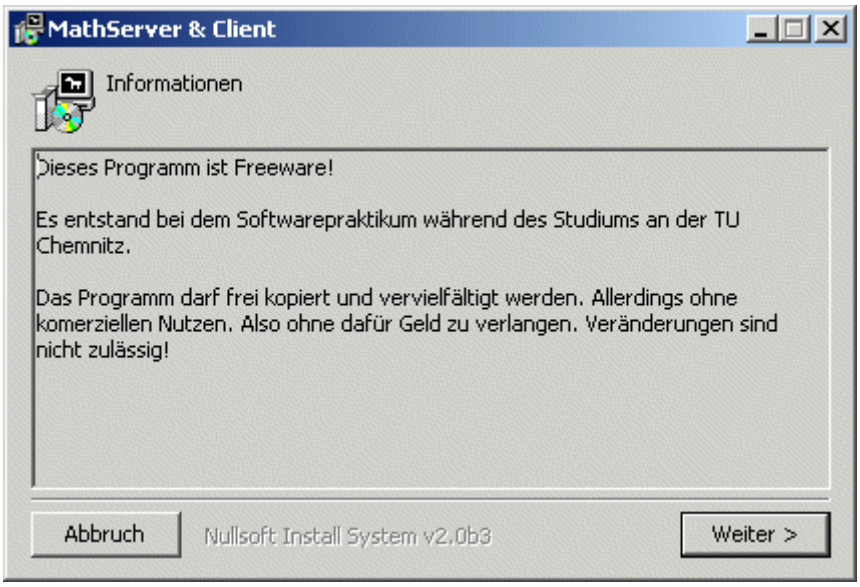
3. Installation

Die Installation läuft in 3 einfachen schritten ab:

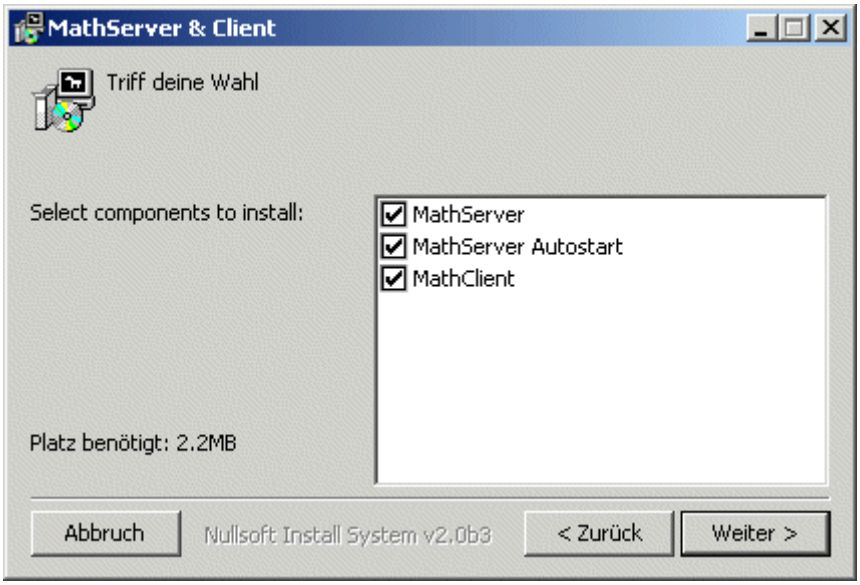
1. CD einlegen und die dort befindliche „Math-Server&Client_v1.0.exe“ starten

2. Einstellungen wie bei den Bildern beschrieben tätigen

3. Fertig


MathServer & Client
Informationen
Dieses Programm ist Freeware!
Es entstand bei dem Softwarepraktikum während des Studiums an der TU Chemnitz.
Das Programm darf frei kopiert und vervielfältigt werden. Allerdings ohne komerziellen Nutzen. Also ohne dafür Geld zu verlangen. Veränderungen sind nicht zulässig!
Abbruch
Nullsoft Install System v2,0b3
Weiter >

Anzeige der Nutzungsbedingungen unter denen das Programm benutzt werden darf

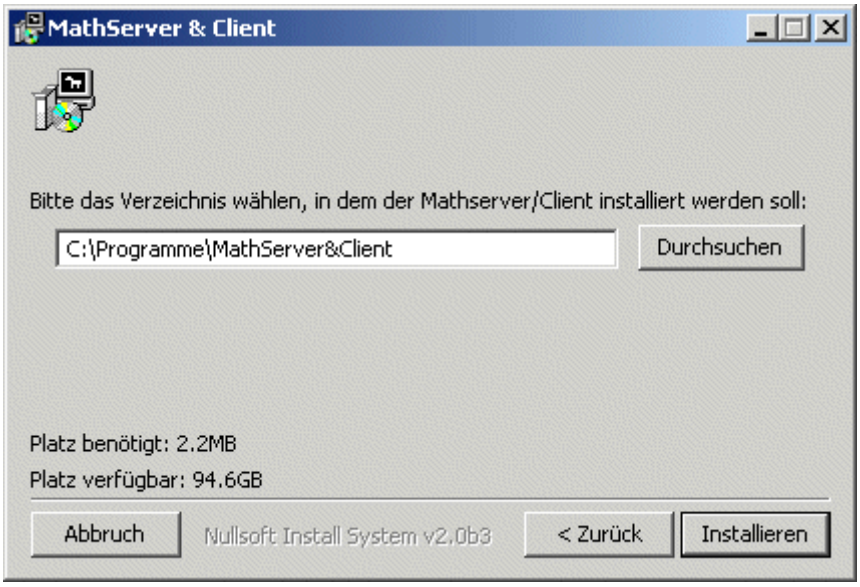


Wollen sie den Server und den Klienten installieren um einen besseren Taschenrechner zu haben oder den Server automatisch beim Computerstart aufrufen? Treffen Sie hier ihre Wahl.

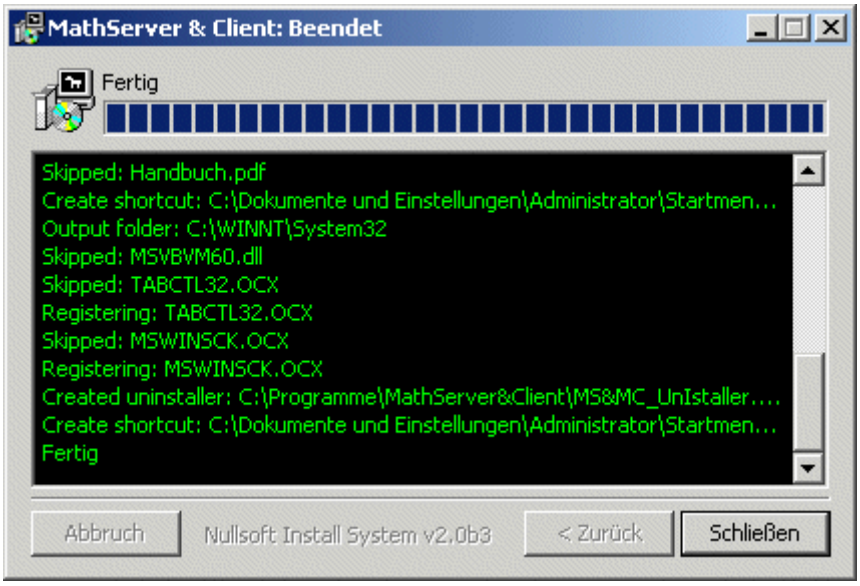
Die erste Checkbox installiert den Mathserver.

Die Zweite richtet den MathServer-Autostart ein.

Die Dritte Installiert den MathClient.



Geben Sie hier den Pfad an, in dem Sie das Programm installieren wollen. Standardmäßig auf C:\Programme\MathServer&Client eingestellt.



Diese Anzeige erscheint wenn die Installation abgeschlossen ist. Die Installation wird mit „Schließen“ beendet.

4. MathServer

4.1 Bedienung

MathServer wird durch „MathServer.exe“ oder den Link im „Startmenü/Programme/MathServer & Client“ gestartet, wenn bei der Installation kein Autostart gewählt wurde und erwartet Eingabedaten in Form eines HTTP-Streams der in XML-Form speziell MathML gestaltet ist.

Extensible Markup Language (XML) 1.0 (Second Edition)	http://www.w3.org/TR/REC-xml
Mathematical Markup Language (MathML) Version 2_0	http://www.w3.org/TR/MathML2

Der MathServer kann nur über den Taskmanager (Strg+Shift+Esc) unter Prozesse beendet werden.

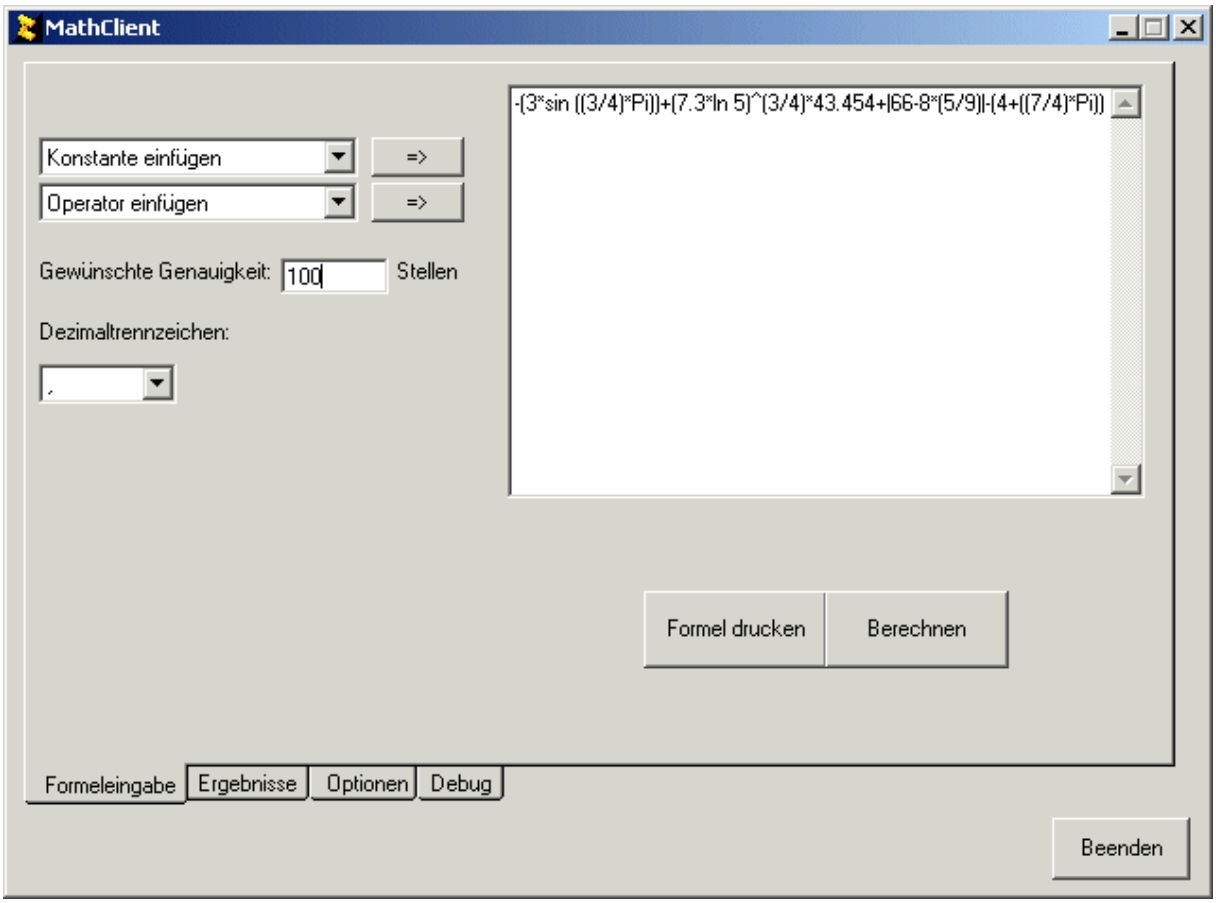
4.2 Unterstützte Funktionen

Siehe MathClient und/oder MathML Dokumentation.

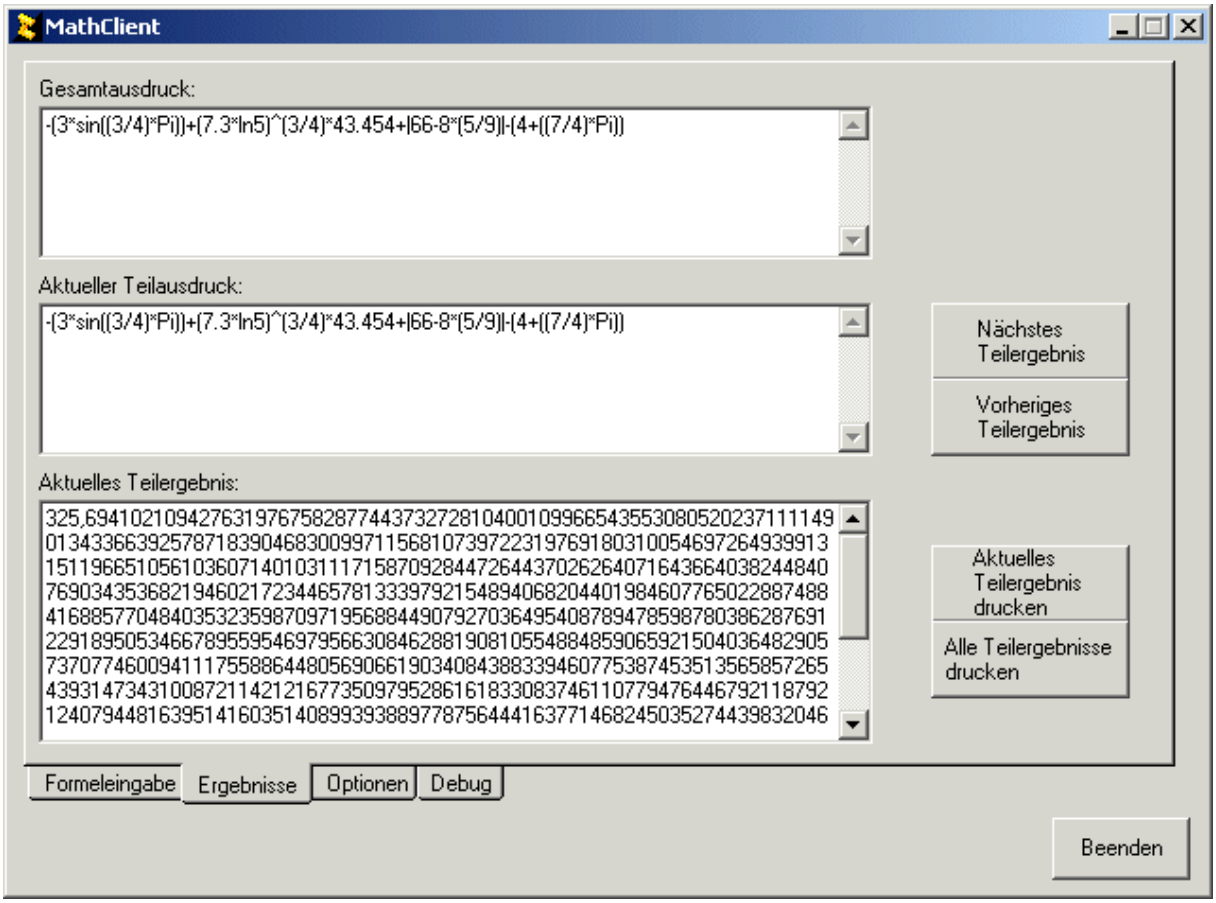
5. MathClient

5.1 Bedienung

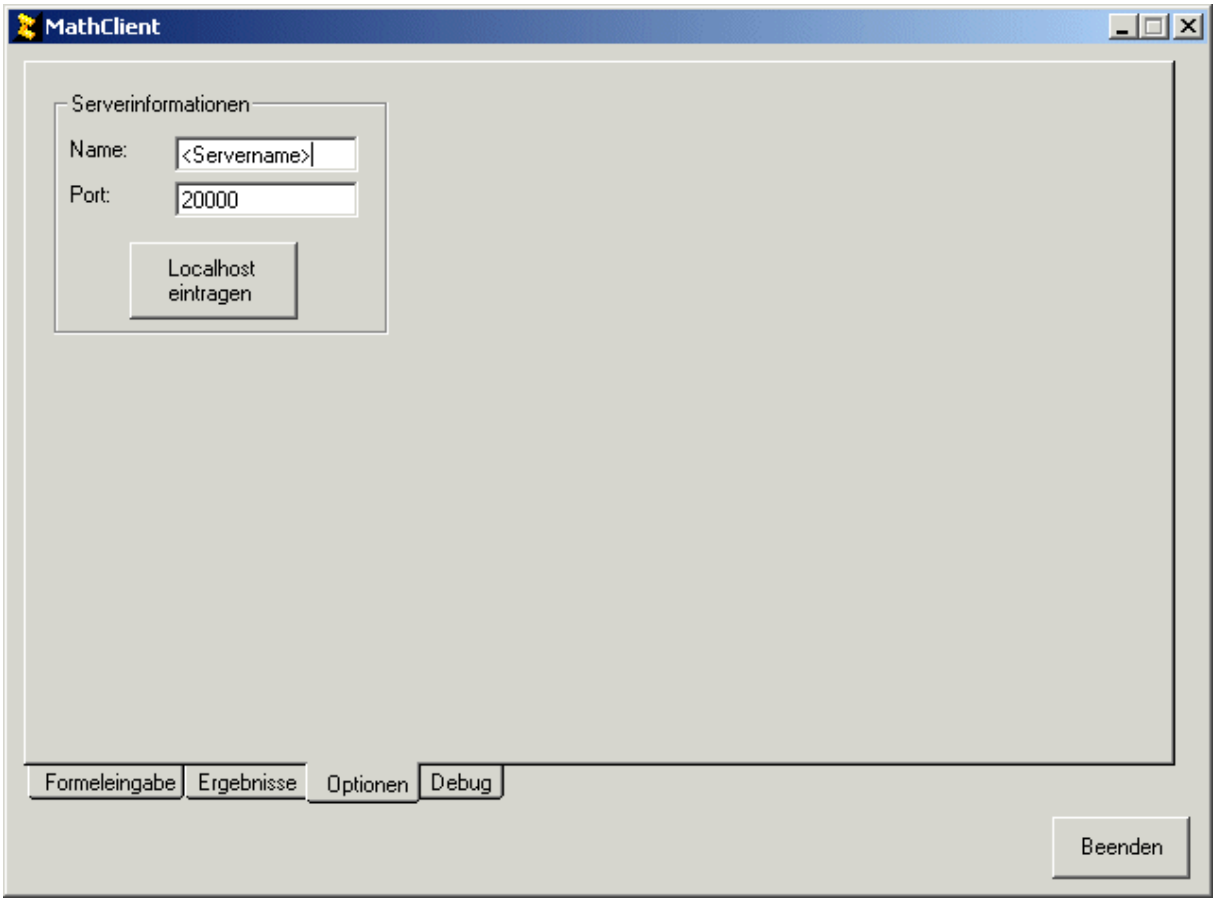
Der Klient ist aus 4 Seiten aufgebaut welche sich durch die 4 Schaltflächen unterhalb der Operationsfläche umschalten lassen.



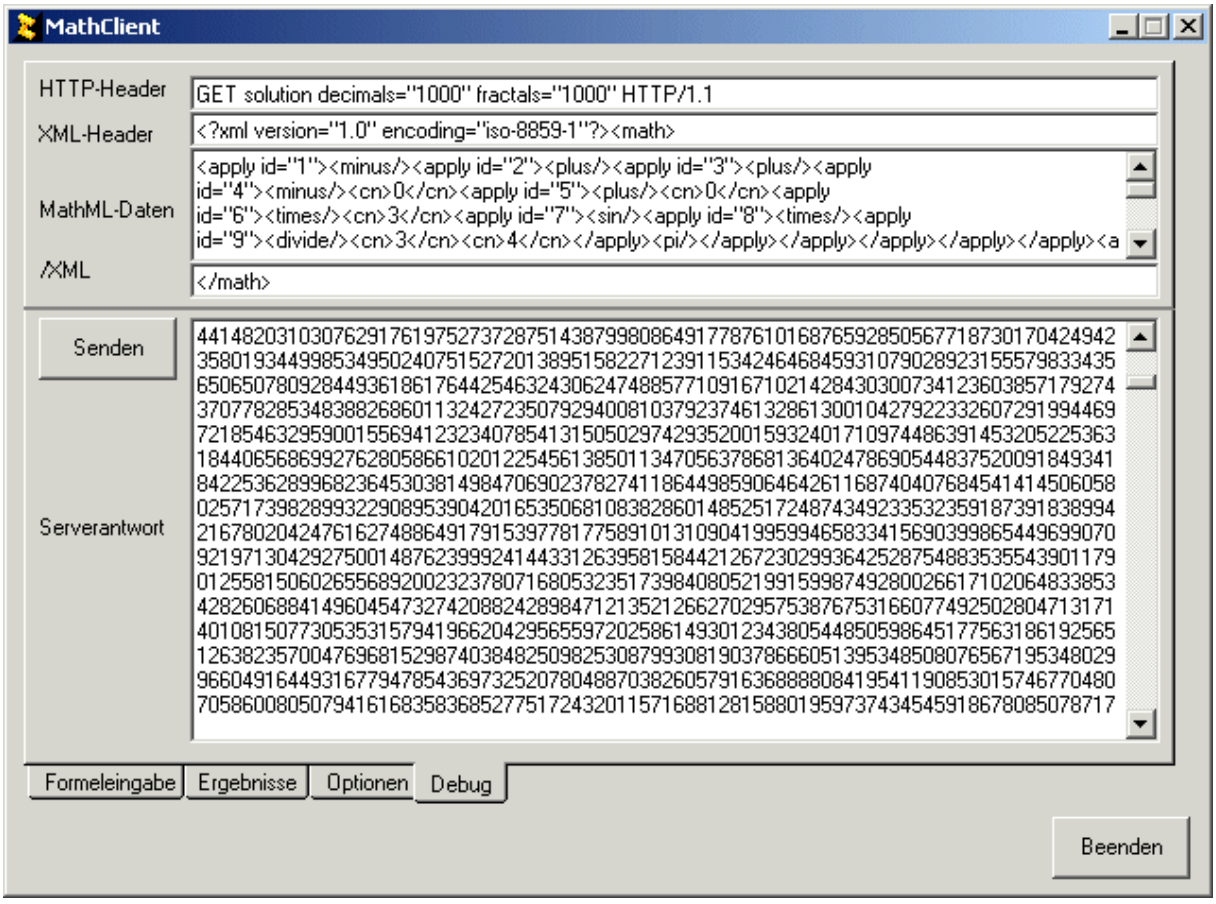
Diese Seite dient zum eingeben der Formel mit den beiden Comboboxen für Konstanten und Operatoren und einer Eingabe für die gewünschte Genauigkeit mit der gerechnet werden soll. Sie ist gleichzeitig Vor- und Nachkommastellenanzahl mit einem Maximum von 100 Mio. Alle unterstützten Funktionen und Konstanten sind in den Comboboxen enthalten. Desweiteren kann man wählen, welches Dezimaltrennzeichen man verwenden möchte.
Mit „Formel drucken“ kann man die eingegebene Formel ausdrucken lassen und „Berechnen“ schickt die Aufgabe zum Server um sie berechnen zu lassen.



Im obersten Fenster erschein nochmals Ihre gestellte Aufgabe, darunter der Teilausdruck, wessen Ergebnis im letzten Fenster angezeigt wird. Standardmäßig die des Gesamtausdrucks.
Durch „Nächstes Teilergebnis“ und „Vorheriges Teilergebnis“ können die Teilergebnisse selektiert werden. Diese können auch einzeln oder komplett ausgedruckt werden durch die Buttons „Aktuelles Teilergebnis drucken“, „Alle Teilergebnisse drucken“.



Diese Seite dient zum einstellen der Netzwerkverbindung. Anstatt <Servername> muss der Computername oder die IP des Computers angegeben werden auf dem der MathServer gestartet wurde. Der Port sollte auf dem Standardwert von 20000 belassen werden, um beste Performance zu erzielen.



Nur erfahrene Nutzer sollten diese Seite verwenden. Hier ist es möglich MathML-Strings per Hand einzugeben und zu verschicken. Sie dient hauptsächlich zu Debugzwecken (Fehlersuche).

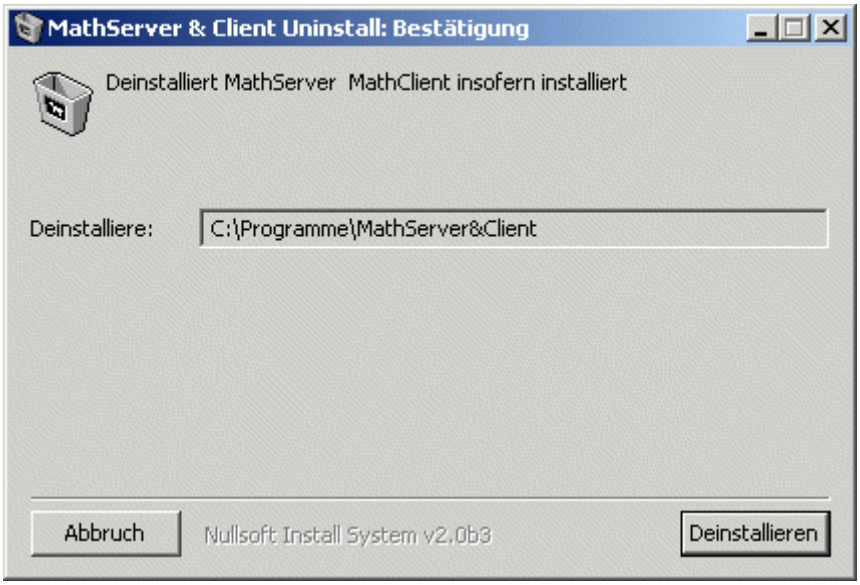
5.2 Unterstütze Funktionen

Funktionen		Beispiel	Konstanten
Addition	+	5+10	
Allgemeiner Logarithmus	log	5log10	EULERsche Zahl (e)
Arkussinus	arcsin	arcsin(0,5)	
Arkustangens	arctan	arctan(5)	LUDOLFsche Zahl oder Kreiskonstante (pi)
Betrag	|	|5|	
Division	/	10/5	
Fakultät	!	5!	
Kosinus	cos	cos(0,5)	
Multiplikation	*	5*10	
Natürlicher Logarithmus	ln	ln5	
Potenz	^	5^10	
Sinus	sin	sin(0,5)	
Subtraktion	-	10-5	
Tangens	tan	tan(5)	
Wurzel	root	root(5)	

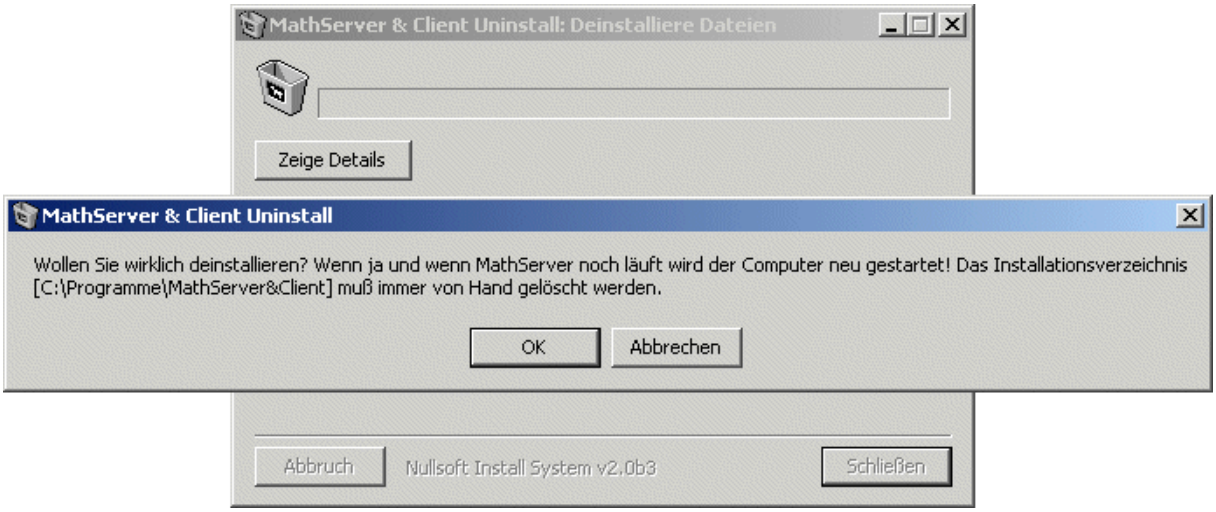
6. Deinstallation

Falls Sie sich wirklich dazu entschließen sollten, dieses wundervolle Programm zu deinstallieren, dann tun Sie das über den entsprechenden Startmenüeintrag. Die Deinstallation läuft dann wie auf den Bildern ab.

In diesem Bild wird IHR Installationspfad angegeben.



Das Installationsverzeichnis des Programms muss von Hand gelöscht werden!
Wenn der MathServer nicht läuft erfolgt kein Neustart.


MathServer & Client Uninstall: Deinstalliere Dateien
Zeige Details
MathServer & Client Uninstall
Wollen Sie wirklich deinstallieren? Wenn ja und wenn MathServer noch läuft wird der Computer neu gestartet! Das Installationsverzeichnis [C:\Programme\MathServer&Client] muß immer von Hand gelöscht werden.
OK
Abbrechen
Abbruch
Nullsoft Install System v2.0b3
Schließen